FAST ICA FOR BLIND SOURCE SEPARATION AND ITS IMPLEMENTATION

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF TECHNOLOGY

IN

VLSI DESIGN AND EMBEDDED SYSTEM

BY

SASMITA KUMARI BEHERA

Department of Electronics and Communication Engineering

National Institute of Technology

Rourkela-769008

2009

FAST ICA FOR BLIND SOURCE SEPARATION AND ITS IMPLEMENTATION

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF TECHNOLOGY

IN

VLSI DESIGN AND EMBEDDED SYSTEM

BY

SASMITA KUMARI BEHERA

Under the guidance of

Prof. Ganapati Panda

Department of Electronics and Communication Engineering

National Institute of Technology

Rourkela-769008

2009

National Institute of Technology
Rourkela

CERTIFICATE

This is to certify that the thesis entitled, "**Fast ICA for Blind Source Separation and its Implementation**" submitted by Miss **Sasmita Kumari Behera** in partial fulfillment of the requirements for the award of Master of Technology Degree in **Electronics and Communication Engineering** with specialization in "**VLSI Design and Embedded System**" during 2008-2009 at the National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by her under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University/ Institute for the award of any degree or diploma.

Date: 15/05/09

Prof. G. Panda (FNAE, FNASc**)**
Dept. of Electronics & comm. Engineering
National Institute of Technology, Rourkela
Rourkela - 769008
Orissa, India

ACKNOWLEDGEMENTS

First of all, I would like to express my deep sense of respect and gratitude towards my advisor and guide **Prof. Ganapati Panda,** who has been the guiding force behind this work. I am greatly indebted to him for his constant encouragement, invaluable advice and for propelling me further in every aspect of my academic life. His presence and optimism have provided an invaluable influence on my career and outlook for the future. I consider it my good fortune to have got an opportunity to work with such a wonderful person.

Next, I want to express my respects to **Prof. S.K. Patra, Prof. K.K.Mohapatra, Prof. G. S. Rath, Prof. D.P.Acharya , Prof. S.K.Behera,** and **Dr. S. Meher** for teaching me and also helping me how to learn. They have been great sources of inspiration to me and I thank them from the bottom of my heart.

I would like to thank all faculty members and staff of the Department of Electronics and Communication Engineering, N.I.T. Rourkela for their generous help in various ways for the completion of this thesis.

I would also like to mention the name of **Jitendra Kumar Das** for helping me a lot during the thesis period.

I would like to thank all my friends and especially my classmates for all the thoughtful and mind stimulating discussions we had, which prompted us to think beyond the obvious. I've enjoyed their companionship so much during my stay at NIT, Rourkela.

I am especially indebted to my parents for their love, sacrifice, and support.

Sasmita Kumari Behera

Roll No: 207EC210

CONTENTS

ABSTRACT

Independent Component Analysis (ICA) is a statistical signal processing technique having emerging new practical application areas, such as blind signal separation such as mixed voices or images, analysis of several types of data or feature extraction. Fast independent component analysis (Fast ICA) is one of the most efficient ICA technique. Fast ICA algorithm separates the independent sources from their mixtures by measuring non-gaussian. Fast ICA is a common method to identify aircrafts and interference from their mixtures such as electroencephalogram (EEG), magnetoencephalography (MEG), and electrocardiogram (ECG). Therefore, it is valuable to implement Fast ICA for real-time signal processing. In this thesis, the Fast ICA algorithm is implemented by hand coding HDL code. In addition, in order to increase the number of precision, the floating point (FP) arithmetic units are also implemented by HDL coding.To verify the algorithm, MATLAB simulations are also performed for both off line signal rocessing and real-time signal processing.

LIST OF FIGURES/TABLE

CHAPTER

1

INTRODUCTION

1.1 MOTIVATION

This thesis addresses the problem of blind signal separation (BSS) using Fast independent component analysis (Fast ICA). In blind signal separation, signals from multiple sources arrive simultaneously at the receiver array, so that each receiver array output contains a mixture of source signals. Sets of receiver outputs are processed to recover the source signals or to identify the mixing system. The term blind refers to the fact that no explicit knowledge of source signals or mixing system is available. Independent component analysis approach uses statistical independence of the source signals to solve the blind signal separation problems. Application domains include communications, biomedical, audio, image, and sensor array signal processing.

Fast ICA algorithm improves the efficiency of independent component analysis. However, most of the publication focused on offline signal processing using Fast ICA algorithm. It can not be applied to real-time applications such as speech signal enhancement and EEG/MEG essential features extraction for brain computer interface (BCI). In order to realize the real-time signal processing, the Fast ICA algorithm can be implemented on a field-programmable gate array (FPGA) to speed up the computations involved.

1.2 SCOPE OF THE THESIS

For real-time signal processing, the Fast ICA algorithm can be implemented on a field programmable gate array (FPGA).The focus of this project is to implement the hardware Fast ICA by hand coding HDL code. Though there is software that translates the high-level languages such as C code, MATLAB, and even Simulink into HDL code, hand coding gives the implementation not only better performance but also less consumption of gate array in the

FPGA. In addition, the numbers precision in hardware Fast ICA is increased by implementing the hardware floating point (FP) arithmetic by hand coding HDL code. The FP arithmetic allows numbers to be represented in a wider range than fixed point arithmetic. The hardware of implementation is partitioned into modules to simplify the implementation.

1.3 THESIS ORGANIZATION

This thesis is organized as follows:

In **chapter 2**, the basic principle behind the independent component analysis (ICA) technique is discussed. Various contrast functions available for ICA are described. Different existing algorithms for ICA are briefly illustrated. Finally the application domain of ICA technique is presented.

In **chapter 3**, background of Fast ICA algorithm is described. Different normalization approaches of Fast ICA algorithm are discussed. Finally the properties of Fast ICA algorithm are presented in this chapter.

In **chapter 4**, implementation of FP arithmetic is presented.

In **chapter 5**, implementation of Fast ICA algorithm is presented.

In **chapter 6**, simulation results are given. The results include simulations done in both MATLAB and VHDL. Comparative performance in terms of CPU time for deflation approach and symmetric approach of Fast ICA algorithm is also given in this chapter.

Finally we conclude in **chapter 7**.

CHAPTER 2

INDEPENDENT COMPONENT ANALYSIS

2.1 Introduction

Independent component analysis (ICA) is a well-known method of finding latent structure in data. ICA is a statistical method that expresses a set of multidimensional observations as a combination of unknown latent variables. These underlying latent variables are called sources or independent components and they are assumed to be statistically independent of each other. The ICA model is

$$x = f(\theta, s) \qquad (2.1)$$

where $x = (x_1, \ldots, x_m)$ is an observed vector and f is a general unknown function with parameters θ that operates on statistically independent latent variables listed in the vector $s = (s_1, \ldots, s_n)$.A special case of (2.1) is obtained when the function is linear, and we can write

$$x = As \qquad (2.2)$$

where A is an unknown $m \times n$ mixing matrix. In Formulae (2.1) and (2.2) we consider x and s as random vectors. When a sample of observations $X = (x_1, \ldots, x_N)$ becomes available, we write $X = AS$ where the matrix X has observations x as its columns and similarly the matrix S has latent variable vectors s as its columns. The mixing matrix A is constant for all observations.

If both the original sources S and the way the sources were mixed are all unkown, and only mixed signals or mixtures X can be measured and observed, then the estimation of A and S is known as blind source separation (BSS) problem.

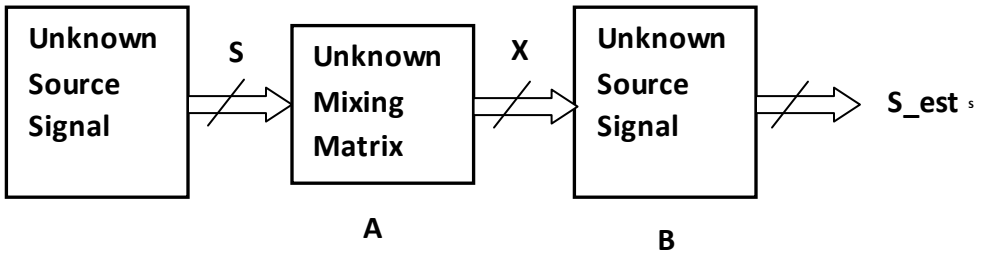

Fig.1. BSS using ICA algorithm

The linear model (2.2) is identifiable under the following fundamental restrictions : the sources (i.e. the components of S) are statistically independent, at most one of the independent components s_j may be Gaussian, and the mixing matrix A must be of full column rank. The identifiability of the model is proved in the case n = m and for those source densities whose variance is defined.

ICA Definition : *"Independent Component Analysis (ICA) is a method for finding underlying factors or components from multivariate (multi-dimensional) statistical data. What distinguishes ICA from other methods is that it looks for components that are both statistically independent and non-Gaussian."*

2.2 Statistical Independence

Statistical independence is the key foundation of independent component analysis. For the case of two different random variables x and y , x is independent of the value of y if knowing the value of y does not give any information on the value of x . Statistical independence is defined mathematically in terms of the probability densities as: the random variables and are said to be independent if and only if

$$P_{x,y}(x,y) = P_x(x)\,P_Y(y) \qquad (2.3)$$

where $P_{x,y}(x,y)$ is the joint density of x and y, $P_x(x)$ and $P_y(y)$ are marginal probability densities of x and y respectively. Marginal probability density function of x is defined as

$$P_x(x) = \int P_{x,y}(x,y)dy \qquad (2.4)$$

Considering a random vector $s = [s_1, s_{2,\ldots\ldots}, s_N]^T$ with multivariate density P(s), has statistically independent components if the density can be factorized as

$$P(s) = \prod_{i=1}^{N} P_i(s_i) \qquad (2.5)$$

Otherway stated, the density of s_1 is unaffected by s_2 ,when two variables s_1 and s_2 are independent. Statistical independence is a much stronger property than uncorrelatedness which takes into account the second order statistics only. If the variables are independent, they are uncorrelated but the converse is not true.

2.3 Central Limit Theorem

The central limit theorem is the most popular theorem in statistical theory and plays an important role in ICA. According to it let

$$x_k = \sum_{i=1}^{k} z_i \qquad (2.6)$$

be a partial sum of sequence $\{Z_i\}$ of independent and identically distributed random variables Z_i. Since the mean and variance of X_k can grow without bound as $k \to \infty$, consider the standardized variables Y_k instead of X_k,

$$Y_k = \frac{X_k - m_{X_k}}{\sigma_{X_k}} \qquad (2.7)$$

where m_{X_k} and σ_{X_k} are mean and variance of X_k .The distribution of Y_k converges to a Gaussian distribution with zero mean and unit variance when $K \to \infty$.

This theorem has a very crucial role in ICA and BSS. A typical mixture or component of the data vector x is of the form

$$x_i = \sum_{j=1}^{N} a_{ij} s_j \qquad (2.8)$$

where a_{ij}, j = 1,2,….N are constant mixing coefficients and s_j, j = 1,2,…N are the N unknown source signals. The central limit theorem can be stated as "the sum of even two independent identically distributed random variables is more Gaussian than the original random variables". This implies that independent random variables are more non-gaussian than their mixtures. Hence non-gaussianity is independence.

2.4 Contrast Functions for ICA

The data model for independent component analysis is estimated by formulating an objective function and then minimizing or maximizing it. Such a function is often called a contrast function or cost function or objective function. The optimization of the contrast function enables the estimation of the independent components. The ICA method combines the choice of an objective function and an optimization algorithm. The statistical properties like consistency, asymptotic variance, and robustness of the ICA technique depend on the choice of the objective function and the algorithmic properties like convergence speed, memory requirements, and numerical stability depend on the optimization algorithm. The contrast function in some way or other is a measure of independence. In this section different measures of independence is discussed which is frequently used as contrast functions for ICA.

2.4.1 Measuring Nongaussianity

1. Kurtosis:

According to the Central limit theorem, nongaussianity is a strong measure of independence. Traditional higher order statistics uses kurtosis or the named fourth-order cumulant to measure non-gaussianity. The kurtosis of a zero-mean random variable v is defined by

$$kurt(v) = E\{v^4\} - 3\left(E\{v^2\}\right)^2 \qquad (2.9)$$

Where $E\{v^4\}$ = fourh moment of v

$E\{v^2\}$ = second moment of v

For a gaussian random variable v, the $E\{v^4\}$ equals $3(E\{v^2\})^2$, so that the kurtosis is zero for a gaussian random variable. If v is a non-gaussian random variable, its kurtosis is either positive or negative. Particularly when kurtosis value is positive the random variables are called *supergaussian* or *leptokurtic* and when negative called *subgaussian* or *platykurtic*. Supergaussian random variables have a _spiky' probability density function with heavy tails and subgaussian random variables have a flat probability density function. Thereore, non-gaussianity is measured by the absolute value of kurtosis.

2. Negentropy:

A second very important measure of nongaussianity is given by negentropy. Negentropy is based on the information-theoretic quantity of (differential) entropy. The entropy of a random variable can be interpreted as the degree of information that the observation of the variable gives. The more random, i.e. unpredictable and unstructured the variable is, the larger its entropy. A fundamental result of information theory is that *a gaussian variable has the largest entropy among all random variables of equal variance*. This means that entropy could be used as a measure of nongaussianity. In fact, this shows that the gaussian distribution is the "most random" or the least structured of all distributions. The differential entropy H of a random variable v with a density of $f(v)$ is given by

$$H(v) = -\int f(v) \log f(v) dv \qquad (2.10)$$

To obtain a measure of nongaussianity that is zero for a gaussian variable and always nonnegative, one often uses a slightly modified version of the definition of differential entropy, called negentropy. Negentropy J is defined as follows

$$J(v) = H(v_{Gauss}) - H(v) \qquad (2.11)$$

where v_{Gauss} is a Gaussian random variable of the same covariance matrix as v. Negentropy is always non-negative, and it is zero if and only if y has a Gaussian distribution.

The advantage of using negentropy as a measure of nongaussianity is that it is well justified by statistical theory. In fact, negentropy is in some sense the optimal estimator of nongaussianity, as far as statistical properties are concerned. The problem in using negentropy is, however, that it is computationally very difficult.

Approximations of negentropy

The estimation of negentropy is very difficult. In practice, some approximation have to be used. These approximation were based on the maximum-entropy principle. In general we obtain the following approximation:

$$H(y) = K\left[E\{G(y_i)\} - E\{G(v)\}\right]^2 \qquad (2.11)$$

where K is some positive constants, and v is a Gaussian variable of zero mean and unit variance.

The variable *y* is assumed to be of zero mean and unit variance, and the functions G is some nonquadratic functions.

choosing *G* wisely, one obtains approximations of negentropy that are much better. If *G* is choosen such that it does not grow too fast, one obtains more robust estimators. The following choices of *G* have proved very useful:

$$G_1(v) = \frac{1}{a_1}\log\cosh a_1 v \qquad (2.12)$$

$$G_2(v) = -\frac{1}{a_2}\exp\left(-a_2 v^2/2\right) \qquad (2.13)$$

$$G_3(v) = \frac{1}{4}v^4 \qquad (2.14)$$

Where a_1 and a_2 are some suitable constants.

2.4.2 Minimization of Mutual Information

Another approach for ICA estimation, inspired by information theory, is minimization of mutual information.

Mutual Information:

Mutual information is a natural measure of dependency between random variables, i.e. it is a measure of the information that member of a set of random variables have on the other random variable in the set. It is always non-negative, and zero if and only if the variables are statistically independent.

If *y* is a n-dimensional random variable and *f(y)* its probability density function then vector *y* has mutually independent components if and only if

$$f(y) = f(y_1) \bullet f(y_2) \bullet ....f(y_n) \qquad (2.16)$$

A natural way of checking whether y has ICs is to measure a distance between both sides of above equation

$$I(f(y)) = \delta(f(y), \prod f(y_i)) \qquad (2.17)$$

Average mutual information of y is given by

$$I(f(y)) = \int f(y) \log\left(\frac{f(y)}{\prod f(y)}\right) dy \qquad (2.18)$$

Average mutual information vanishes if and only if the variables are mutually independent and strictly positive otherwise. In terms of negentropy mutual information is written as

$$I(y_1, y_2, ..., y_m) = H(y) - \sum_{i=1}^{m} H(y_i) \qquad (2.19)$$

But the contrast functions based on mutual information discussed above require the estimation of the density function and this has severely restricted the use of these contrast functions.

2.5 Data preprocessing for ICA

It is often beneficial to reduce the dimensionality of the data before performing ICA. It might be well that there are only a few latent components in the high-dimensional observed data, and the structure of the data can be presented in a compressed format. Estimating ICA in the original, high-dimensional space may lead to poor results. For example, several of the original dimensions may contain only noise. Also, over learning is likely to take place in ICA if the number of the model parameters (i.e., the size of the mixing matrix) is large compared to the number of observed data points. Care must be taken, though, so that only the redundant dimensions are removed and the structure of the data is not flattened as the data are projected to a lower dimensional space. In this section two methods of dimensionality reduction are discussed: principal component analysis and random projection.

In addition to dimensionality reduction, another often used preprocessing step in ICA is to make the observed signals zero mean and decorrelate them. The decorrelation removes the second-order dependencies between the observed signals. It is often accomplished by principal component analysis which will be briefly described next.

2.5.1 Principal component analysis

In principal component analysis (PCA) , an observed vector **x** is first centered by removing its mean (in practice, the mean is estimated as the average value of the vector in a sample). Then the vector is transformed by a linear transformation into a new vector, possibly of lower dimension, whose elements are uncorrelated with each other. The linear transformation is found by computing the eigenvalue decomposition of the covariance matrix. For a zero-mean vector x, with n elements, the covariance matrix C_x is given by:

$$C_x = E\{xx^T\} = EDE^T \qquad (2.20)$$

Where $E = (e_1, e_2, \ldots, e_n)$ =orthogonal matrix of eigenvectors o C_x

$D = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ = diagonal matrix of eigenvalue of C_x

Whitening or sphering can be described as

$$z = Px \qquad (2.21)$$

where P is the whitening matrix and z is a new matrix that is white.

P is defined as

$$P = D^{-1/2} \times E^T \qquad (2.22)$$

Subsequent ICA estimation is done on z instead of x. For whitened data it is enough to find an orthogonal demixing matrix if the independent components are also assumed white. Dimensionality reduction can also be accomplished by methods other than PCA. These methods include local PCA and random projection.

2.6 Algorithms for ICA

Some of the ICA algorithms require a preprocessing of observed data and some may not. Algorithms those need no preprocessing (centering and whitening), often converge better with whitened data. However in certain cases if it is necessary then sphered data is used otherwise no mention of sphering is done for cases where whitened is not required.

2.6.1 Nonlinear PCA Algorithm

An approach to ICA that is related to PCA is the so called non-linear representation is sought for the input data that minimizes a least mean square error criterion. For linear case principal components are obtained and in some cases the nonlinear PCA approach gives independent components instead. A hierarchical PCA learning rule is given by

$$\Delta w \propto g(y_i)x - g(y_i)\sum_{j=1}^{i} g(y_i)w_j \qquad (2.23)$$

where g is a suitable non-linear scalar function. The introduction of non-linearities means that the learning rule uses higher order information in the learning. It is proven that for well chosen non-linearities, the learning rule in above equation does indeed perform ICA, if the data is whitened.

2.6.2 Non-linear cross correlation based Algorithm:

Principle of cancellation of non-linear cross correlation is used to estimate independent components. Non-linear cross correlations are of the form E{ $g_1(y_i)$, $g_2(y_j)$} where g_1 and g_2 are some suitably chosen nonlinearities. If i and j are independent, then these cross correlations are zero for y_i and y_j having symmetric densities. The objective function in such cases is formulated implicitly and exact objective function may not even exist. Jutten and Herault used this principle to update the nondiagonal terms of the matrix according to

$$\Delta W_{ij} \propto g_1(y_i)g_2(y_j) \text{ for } i \neq j \qquad (2.24)$$

Here y_i are computed at every iteration as $Y = (I + W)^{-1}$ and the diagonal terms W_{ii} are set to zero. After convergence y_i give the estimates of the independent components. However the algorithm converges only under severe restrictions.

2.6.3 Nonlinear Decorrelation Algorithm

To reduce the computational overhead by avoiding matrix inversions in Jutten- Herault algorithm and improve stability some algorithm has been proposed. In those the following algorithm has been proposed

$$\Delta W \propto (I - g_1(y)\, g_2(y^T))\, W \qquad (2.24)$$

where , $y = W x$, the nonlinearities $g_1(.)$ and $g_2(.)$ are applied separately on every component of the vector y, and the identity matrix can be replaced by any positive definite diagonal matrix. According to EASI algorithm,

$$\Delta W \propto (I - y y^T - g(y)\, y^T + y\, g(y^T))\, W \qquad (2.25)$$

The choice of the nonlinearities used in above rules is generally provided by the maximum likelihood (or infomax) approach.

Other ICA algorithms that are available are : Tensor based ICA Algorithm, One-unit neural learning rules, Infomax Estimation or Maximum Likelihood Algorithm, Algebraic ICA Algorithm, Evolutionary ICA Algorithm,and Fast ICA Algorithm.The Fast ICA algorithm is described in detail in the following chapter.

2.7 Applications of ICA

ICA being a blind statistical signal processing technique finds application in many application areas such as blind separation of mixed voices or images, analysis of several types of data , feature extraction , speech and image recognition, data communication ,sensor signal processing, system identification, biomedical signal processing and several others .

Biomedical signals such as electroencephalogram (EEG), magnetoencephalography (MEG), and electrocardiogram (ECG) are generally measured from clinical sensors or instruments; however measured signals are polluted by the aircrafts and other unknown noise signals, such as eye movements, muscle noise, and power noise from instruments. This problem can be solved by independent component analysis(ICA) algorithm, which identifies aircrafts from the measured signals.

Another application area of great potential is **telecommunications**. An example of a real-world communications application where blind separation techniques are useful is the separation of the user's own signal from the interfering other users' signals in CDMA (Code-Division Multiple Access) mobile communications. This problem is semi-blind in the sense that certain additional prior information is available on the CDMA data model. But the number of parameters to be estimated is often so high that suitable blind source separation techniques taking into account the available prior knowledge provide a clear performance improvement over more traditional estimation techniques.

ICA is successfully used for **face recognition**. The goal in face recognition is to train a system that can recognize and classify familiar faces given a different image of the trained face. The test images may show the faces in a different pose or under different lighting conditions. Traditional methods for face recognition have employed PCA-like methods. The rows of the face images

constitute the data matrix x. Performing ICA, a transformation W is learned so that u (u = Wx) represent the independent face images.

A **sensor network** is a very recent, widely applicable and challenging field of research. Multi-sensor data often presents complementary information about the region surveyed and data fusion provides an effective method to enable comparison, interpretation and analysis of such data. Image and video fusion in a sub area of the more general topic of data fusion dealing with image and video data. ICA is also used for robust automatic speech recognition. Applications of ICA also include feature extraction in images and finding hidden factors in financial data.

There are two thoughts with respect to what actually is the aim in estimating the independent components in the data. First, one may regard the data being generated by a combination of some existing but unknown independent source signals s_j, and the task is to estimate them. This viewpoint is chosen in the so called blind source separation (BSS) framework — there are some sources which have been mixed, and the mixing process is completely unknown to us (hence the word "blind"). The application areas of ICA listed above mostly fall into the BSS category.

Another point of view is to regard ICA as a method of presenting the data in a more comprehensible way by revealing the hidden structure in the data and often reducing the dimensionality of the representation. According to this latter school of thought, it might well be that there are no "true" source signals generating the data — it still pays to represent the data as a combination of a few latent factors that are statistically as independent as possible. This view can be called a data mining approach of the problem

CHAPTER 3

FAST ICA ALGORITHM

3.1 Inroduction

In the previous chapter, different measures of nongaussianity are introduced, i.e. objective functions for ICA estimation. In practice, one also needs an algorithm for maximizing the contrast function. In this section, we introduce a very efficient method of maximization suited for this task.

3.2 Fast ICA for One Independent Component

Assume that we have collected a sample of the sphere (or prewhitened) random vector x,which is in case of blind source separation is a collection of linear mixture of independent source signals. The basic method of Fast ICA algorithm is as follows:

1. Take a random initial vector w(0) and divide it by its norm. Let k = 1 .
2. Let $w(k) = E\{Z[Z^T w(k-1)]^3\} - 3w(k-1)$ (3.1)
3. Divide w(k) by its norm .
4. If $|w^T(k)w(k-1)|$ is not close enough to 1, let k = k+1, and go back to step 2.Otherwise the algorithm is convergent and outputs w(k) .

The final vector w(k) given by the algorithm equals one of the columns of the (orthogonal) demixing matrix B. In case of blind source separation, this means that w(k) separates one of the non-Gaussian source signals in the sense that $w(k)^T x(t)$, t = 1,2,……. equals one of the source signals.

3.3 Fast ICA for Several Independent Components

To estimate n independent components, run these algorithm n times. To ensure that we estimate each time a different independent component, we only need to add a simple orthogonalizing projection inside the loop. The column of the demixing matrix B is orthonormal because of the sphering. Thus we can estimate the independent components one by one by projecting the current solution w(k) on the space orthogonal to the columns of the demixing matrix B previously found. Define the matrix B as the matrix whose columns are the previously found columns of B.

Then adding the projection operation in the beginning of step 3.

3. $w = w - B\overline{B}^{T} \times w$ (3.2)

Divide w(k) by its norm.

Also the initial random vector should be projected this way before starting the iterations. To prevent estimation error in $\overline{B}$ from deteriorating the estimate w(k), this projection step can be omitted after the first few iterations: once the solution w(k) has entered the basin of attraction of one of the fixed points, it will stay there and converge to that fixed point.

In addition to the hierarchical (or sequential) orthogonalization described above, any other method of orthogonalizing the weight vectors could also be used. I n some applications, a symmetric orthogonalization might be useful. This means that the fixed point step is first performed for all the n weight vectors, and then the matrix W(k) = ($w_1(k)$,….. $w_n(k)$) of the weight vector is orthogonalized, e.g., using the well known formula:

$$W(k) = W(k)\,(W(k)^{T} W(k))^{1/2} \qquad (3.3)$$

Where $(W(k)^T W(k))^{1/2}$ is obtained from the eigenvalue decomposition of $W(k)^T W(k) = EDE^T$ as $(W(k)^T W(k))^{1/2} = E D^{1/2} E^T$.

3.4 A Semi-Adaptive Version

A disadvantage of many batch algorithms is that large amount of data must be stored simultaneously in working memory. The fixed point algorithm however, can be used in a semi-adaptive manner so as to avoid this problem. This can simply be accomplished by computing the expectation $E\{x(w(k-1)^T x)^3\}$ by an on-line algorithm for N consecutive sample points, keeping w(k-1) fixed, and updating the vector w(k) after the average over all the N sample points has been computed.

This semi-adaptive version also makes adaptation to non-stationary data possible. Thus the semi-adaptive algorithm combines many of the advantages usually attributed to either on-line or batch algorithms.

3.5 Orthonormalization of Fast ICA Algorithm

The Fast ICA algorithm is an iterative method to find the local maximum of a cost function defined by

$$J_G = \sum_{I=1}^{n} E\{G(W_i^T Z)\} \qquad (3.4)$$

With G an even symmetrical function. The symbol E stands for expectation, which in practice would be estimated by sample mean over the whitened vectors z . A widely used cost function is the fourth-order cumulant or kurtosis, defined for any random variable *v* as

$$kurt(v) = E\{v^4\} - 3\left(E\{v^2\}\right)^2 \qquad (3.5)$$

With the constraint that the argument $w_i^T z = y_i$ has unit variance the cost function becomes

$$J_G^{kurt} = \sum_{i=1}^{n} E\left\{\left(w_i^T z\right)^4\right\} \qquad (3.6)$$

For the one-unit case, in which only one of the rows of $\mathbf{W}$ is considered and orthogonalization is reduced to just normalization of the vector to unit length after each iteration step, the fastICA algorithm for the general cost function (1), the updating step is

$$\overline{w_i} = E\{z g(w_i^T z)\} - E\{g'(w_i^T z)\} w_i \qquad (3.7)$$

With function g the derivative of G and g' the derivative of g. For the kurtosis cost function, the corresponding updating step is

$$\overline{w_i} = E\left\{z\left(w_i^T z\right)^3\right\} - 3 \times w_i \qquad (3.8)$$

To obtain the full matrix **W**, we need to run the one-unit algorithm n times and the vector $\overline{w_i}$ must be reorthonormalized after the update because they lose their orthonormality in the updating step.

The orthonormalization can be accomplished basically in two ways:

1. Deflationary orthonormalization
2. Symmetrical orthonormalization

3.5.1 Deflationary Orthonormalization

The estimated components are obtained one by one in the FastICA algorithm with deflation orthonormalization. Deflationary orthonormalization is given by

$$w_p = w_p - \sum_{j=1}^{p-1} (w_p^T w_j) w_j \qquad (3.9)$$

With p the previously estimated vectors number.

3.5.2 Symmetric Orthonormalization

In certain applications, it may be desirable to use a symmetric decorrelation, in which no vectors are "privileged" over others; This means that the vectors w_i are not estimated one by one; instead, they are estimated in parallel. One motivation for this is that the deflationary method has the drawback that estimation errors in the first vectors are cumulated in the subsequent ones by the orthonormalization. Another one is that the symmetric orthonormalization methods enable parallel computation of independent components.

Symmetric orthonormalization is given by

$$W = \left(\overline{WW}^T\right)^{-1/2} \overline{W} \qquad (3.10)$$

Where $\overline{W}$ is the matrix with rows $\overline{w}_i^T$. This means that the updating step is first performed for all the n weight vectors, and then the matrix **W** is orthogonalized using (3.10).

Symmetric orthonormalization is done by first doing the iterative step of the one-unit algorithm on every vector w_i in parallel, and afterwards orthogonalizing all the w_i by special symmetric methods. In other words:

1) Initialize the w_i, i = 1,2,…..n(sample length)

2) Do an iteration of a one-unit algorithm on every w_i in parallel.

3) Do a symmetric orthogonalization of the matrix **W**.

4) If not converged, go back to step 3.

3.6 Properties of Fast ICA Algorithm

The Fast ICA algorithm and the underlying contrast functions have a number of desirable properties when compared with existing methods for ICA.

1. The convergence is cubic (or at least quadratic), under the assumption of the ICA data model This is in contrast to ordinary ICA algorithms based on (stochastic) gradient descent methods, where the convergence is only linear. This means a very fast convergence
2. Contrary to gradient-based algorithms, there are no step size parameters to choose. This means that the algorithm is easy to use.
3. The algorithm finds directly independent components of (practically) any non-Gaussian distribution using any nonlinearity g. This is in contrast to many algorithms, where some estimate of the probability distribution function has to be first available, and the nonlinearity must be chosen accordingly.

4. The performance of the method can be optimized by choosing a suitable nonlinearity g. In particular, one can obtain algorithms that are robust and/or of minimum variance.
5. The independent components can be estimated one by one, which is roughly equivalent to doing projection pursuit. This is useful in exploratory data analysis, and decreases the computational load of the method in cases where only some of the independent components need to be estimated.
6. The Fast ICA has most of the advantages of neural algorithms: It is parallel, distributed, computationally simple, and requires little memory space. Stochastic gradient methods seem to be preferable only if fast adaptively in a changing environment is required.

CHAPTER 4

IMPLEMENTATION OF FLOATING POINT ARITHMETIC

4.1 Introduction

Many scientific problems require FP arithmetic with high precision in their calculations. Moreover a large dynamic range of numbers is necessary for signal processing.FP arithmetic has the ability to automatically scale numbers and allows numbers to be represented in a wider range than fixed-point arithmetic. Nevertheless, FP algorithm is difficult to implement on the FPGA, because the algorithm is so complex that the area (logic elements) of FPGA leads to excessive consumption when implemented. A simplified 32-bit FP implementation includes adder, subtractor, multiplier, divider, and square rooter.

4.2 Floating Point Representation

The format of IEEE 745 standrd 32-bit FP number is given in Fig 4.1:

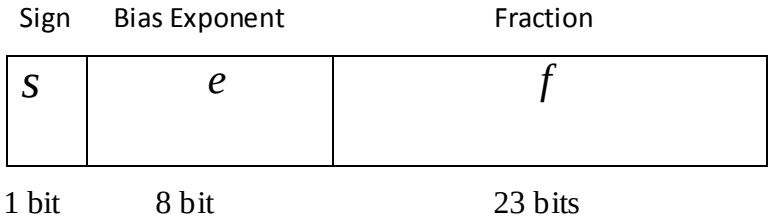

Fig. 4.1 IEEE 745 single precision FP format

In it , *s* is th sign bit used to specify the sign of the FP number, *e* is the 8-bit quantity called the exponent field, and *f* has 23 bits used to store the binary representation of the FP number. The leading one in the mantisa 1.*f* does not appear in the representation; therefore, the leading one is implicit. The FP value of *fp* is computed by

$$fp = (-1)^s (1.f) \times 2^{(127-e)} \qquad (4.1)$$

4.3 Floating Point Adder/Subtractor

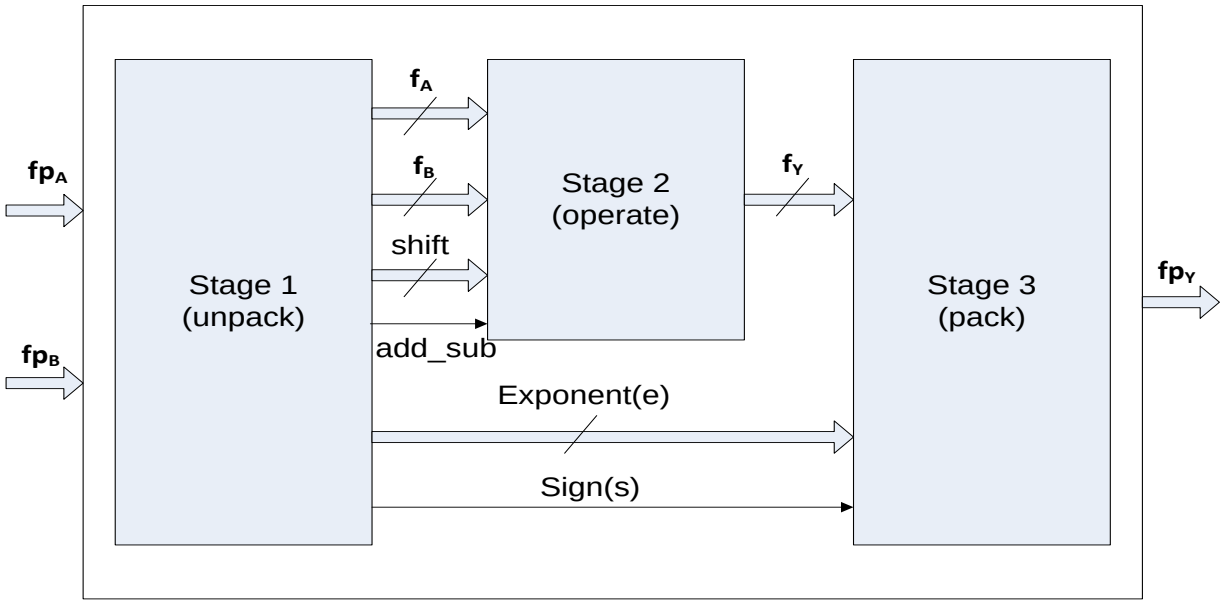

Fig 4.2 Floating point Adder/Subtractor

The block diagram of the FP adder is shown in Fig.4.2 ,which adds FP number fp_A to fp_B. The adder has three stages. The main work of stage 1 is to compare fp_A with fp_B. If fp_A is less than fp_B, swap fp_A and fp_B Stage 2 processes the operations of 1. f_A and 1. f_B. If s_A equals s_B, add 1. f_A to 1. f_B, else subtract 1. f_B from 1. f_A. Stage 3 normalizes the result of stage 2 and then adjusts e_Y . Finally, packs s_Y, e_Y, and f_Y to the output format, and output the adder result fp_Y.

Basic Algorithm:

- Subtract exponent(d = e_A- e_B).
- Align significands.
  - shift right d positions the significand of the operand with the smallest exponent.

- select as the exponent of the result the largest exponent.

- Add(subtract) significands and produce sign of result.
- Normalization of result and adjust the exponent.
- Determine exception flags and special values.

4.4 Floating Point Multiplier

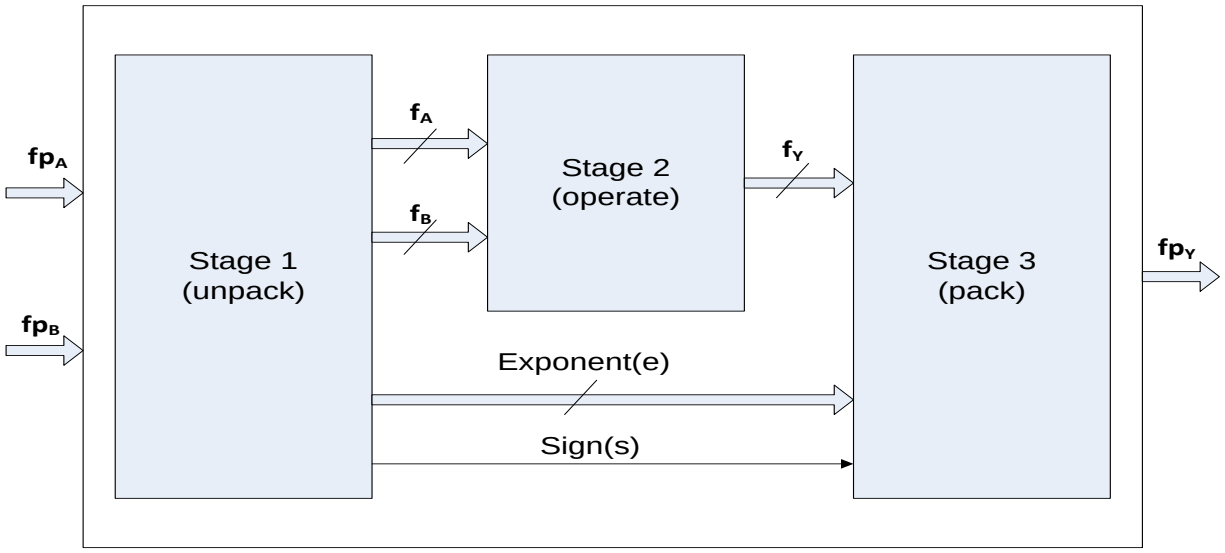

Fig .4.3 Floating point Multiplier

Basic Algorithm:

- Multiply the significands.
- Add the exponents.
- Determine the sign of the result.
- Normalization of result.

- Round.
- Determine the exception flags and special values.

4.5 Floating Point Divider

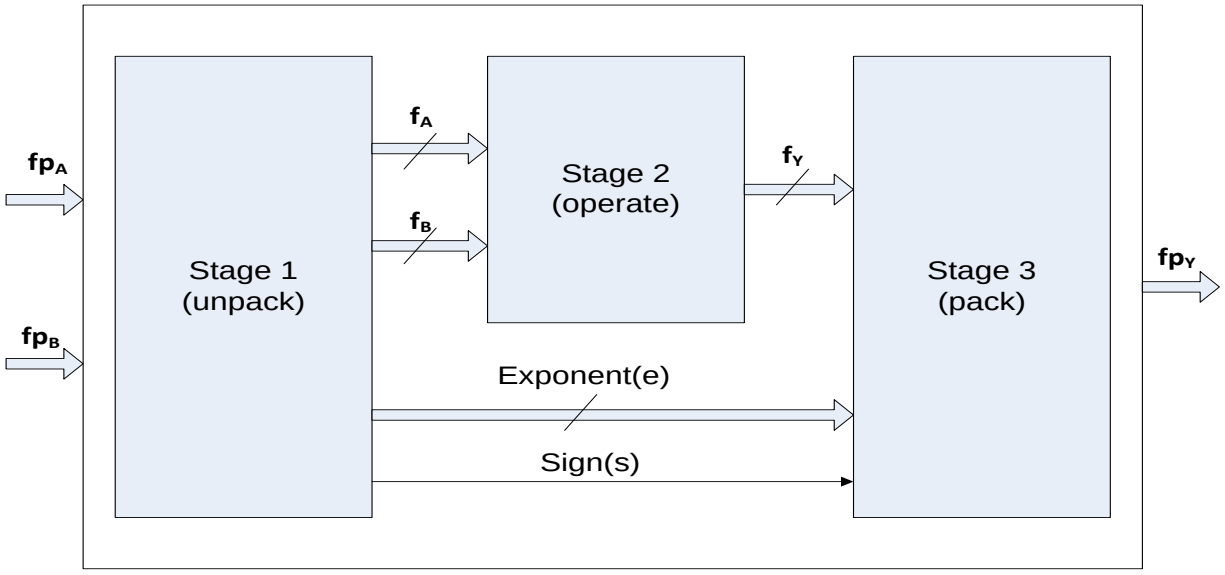

Fig. 4.4 Floating point Divider

FP divider is more complex than FP adder, subtractor, and multiplier.The block diagram of the FP divider is given in Fig 4.4 Stage 1 unpacks fp_i to s_i, e_i, f_i , and then subtracts exponents e_B from e_A. Stage 2 uses the nonrestoring division algorithm to implement the method of $1.f_A$ divided by $1.f_B$. Stage 3, packs s_Y, e_Y, and f_Y to the output format, and output the divider result fp_Y.

Basic Algorithm:

- Divide significands.
- Digit Recurrence Algorithm.
  - Restoring Division
  - Non-Restoring Division
  - SRT Division (Sweeney, Robertson, and Tocher)
- Subtract exponent.
- Normalization of result.
- Round.
- Determine exeption flags and special values.

4.6 Floating point Square Rooter

The block diagram of the FP square rooter is presented in Fig 4.5. Stage 1 unpacks fp_A to s_A, e_A, f_A, and then divides exponent e_A by 2. Stages 2 implement the square rooting of f_A. Stage 3, packs s_Y, e_Y, and f_Y to the output format, and output the square root result fp_Y.

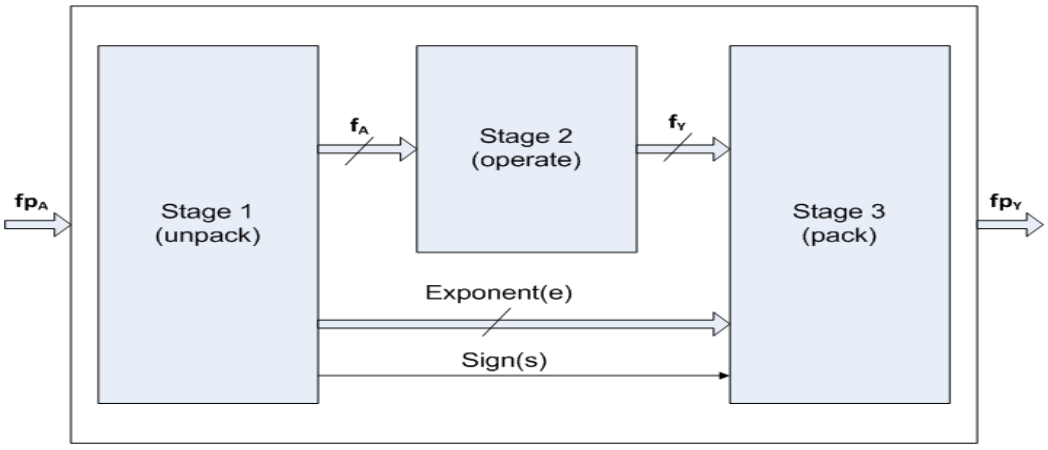

Fig. 4.5 Floating point Square Rooter

Basic Algorithm:

- Obtain the square root of the significand .
- Radix-2 sqare root with carry-save adder.
- produce the exponent of the result.
- Normalize the result and update exponent.
- Round.
- Determine exception flags and special values

CHAPTER

5

IMPLEMENTATION OF FAST ICA ALGORITHM

5.1 Introduction

For real-time applications such as speech signal enhancement, and EEG/MEG essential features extraction for brain computer interface (BCI), the Fast ICA algorithm can be implemented on a field-programmable gate array (FPGA) to speed up the operation. VLSI implementation offers many features such as high processing speed, which is extremely desired in Fast ICA implementations. The complex computation of Fast ICA is one of the main barricades in hardware implementation, especially in synthesis procedure. Therefore, hierarchy and modularity techniques in VLSI design are very much essential. The hierarchy involves dividing an ICA process into subprocessing modules until the complexity of the bottom level submodules becomes manageable. These submodules are independently developed, then integrated together .

5.2 Implementation of Centering

The process of centering is to subtract the mixed signal means μ1 and μ2 from x1 and x2, respectively. First the mixed signal elements are accumulated one by one. After getting the summation of x, μ is obtained by dividing the summation by the sample length. In order to speed up the processing, multiplication operation (multiply by 1/sample length) is used instead of the division. Second, the mean is subtracted from the mixed signal data for achieving centering. The operation is formulated as:

$$x(i) = x(i) - \left(\sum_{j=1}^{samplelength} x(i) \right) \times (1/samplelength) \qquad (5.1)$$

Where i = 1, 2,…..,sample length

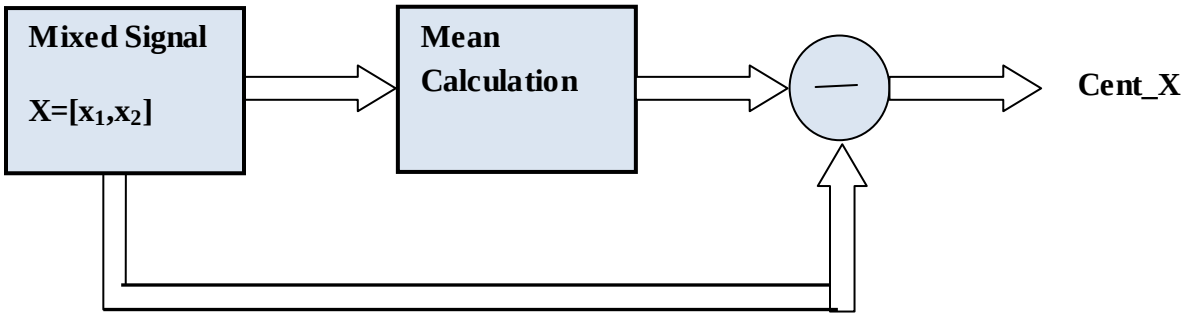

Fig 5.1 Block diagram of implementation of Centering

5.3 Implementation of Whitening

The first step of whitening is to find the whitening matrix **P**. **P** is given by :

$$P = D^{-1/2} \times E^{T} \qquad (5.2)$$

Where D = diag(λ_1, λ_2) = diagonal matrix of the covariance matrix C_X's eigenvalues.

E = (e_1, e_2) = Orthogonal matrix of C_X's eigenvectors.

$C_X = E\{XX^T\}$ is a 2×2 matrix.

It takes three multipliers to implement the calculation of XX^T. Multiplier-1 is used for $x_1 \times x_1$. Multiplier-2 is used for $x_1 \times x_2$, and multiplier-3 is used for $x_2 \times x_2$, where

$$X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1(1), x_1(2), \ldots, x_1(samplelength) \\ x_2(1), x_2(2), \ldots, x_2(samplelength) \end{pmatrix}$$

Because $x_1 \times x_2$ equals $x_2 \times x_1$, it only needs to implement one of them. C_X thus can be derived by multiplying summation by 1/sample length. This operation can be formulated as

$$C_X = \begin{bmatrix} C_{x\_00} & C_{x\_01} \\ C_{x\_10} & C_{x\_11} \end{bmatrix}$$

$$C_{x\_00} = \left(\sum_{j=1}^{samplelength} X_1(j) \times X_1(j) \right) \times (1/samplelength) \tag{5.3}$$

$$C_{x\_01} = C_{x\_10} = \left(\sum_{j=1}^{samplelength} X_1(j) \times X_2(j) \right) \times (1/samplelength) \tag{5.4}$$

$$C_{x\_11} = \left(\sum_{j=1}^{samplelength} X_2(j) \times X_2(j) \right) \times (1/samplelength) \tag{5.5}$$

Once the covariance matrix is calculated, the next step is to determine the orthogonal matrix of eigenvectors of C_x (E) and the diagonal matrix of eigenvalues of C_x (D).

$$\sigma = \frac{C_{x\_22} - C_{x\_11}}{2 \times C_{x\_12}} \tag{5.6}$$

$$T = \frac{sign(\sigma)}{abs(\sigma) + \sqrt{1 + (\sigma \times \sigma)}} \tag{5.7}$$

$$C = \frac{1}{\sqrt{1 + (T \times T)}} \tag{5.8}$$

$$S = T \times C \tag{5.9}$$

$$\mathrm{E} = \begin{bmatrix} C & S \\ -S & C \end{bmatrix} \tag{5.10}$$

$$\mathrm{D} = \mathrm{E}^{\mathrm{T}} \times [\, C_X \times \mathrm{E} \,] \tag{5.11}$$

$$\text{If } \mathrm{D} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \quad \text{then} \quad \mathrm{D}^{-1/2} = \begin{bmatrix} \lambda_1^{-1/2} & 0 \\ 0 & \lambda_2^{-1/2} \end{bmatrix} \tag{5.12}$$

$$\text{where} \quad \lambda_1^{-1/2} = \frac{1}{\sqrt{\lambda_1}} \quad \text{and} \quad \lambda_2^{-1/2} = \frac{1}{\sqrt{\lambda_2}}$$

The whitening matrix P is thus obtained by multiplying $\mathrm{D}^{-1/2}$ by E^{T}

$$\mathrm{P} = \mathrm{D}^{-1/2} \times \mathrm{E}^{\mathrm{T}} \tag{5.13}$$

Finally, the white data Z is obtained after multiplying **P** by **X**.

$$\mathrm{Z} = \mathrm{PX} \tag{5.14}$$

The implementation block diagram is presented in Fig 5.2.

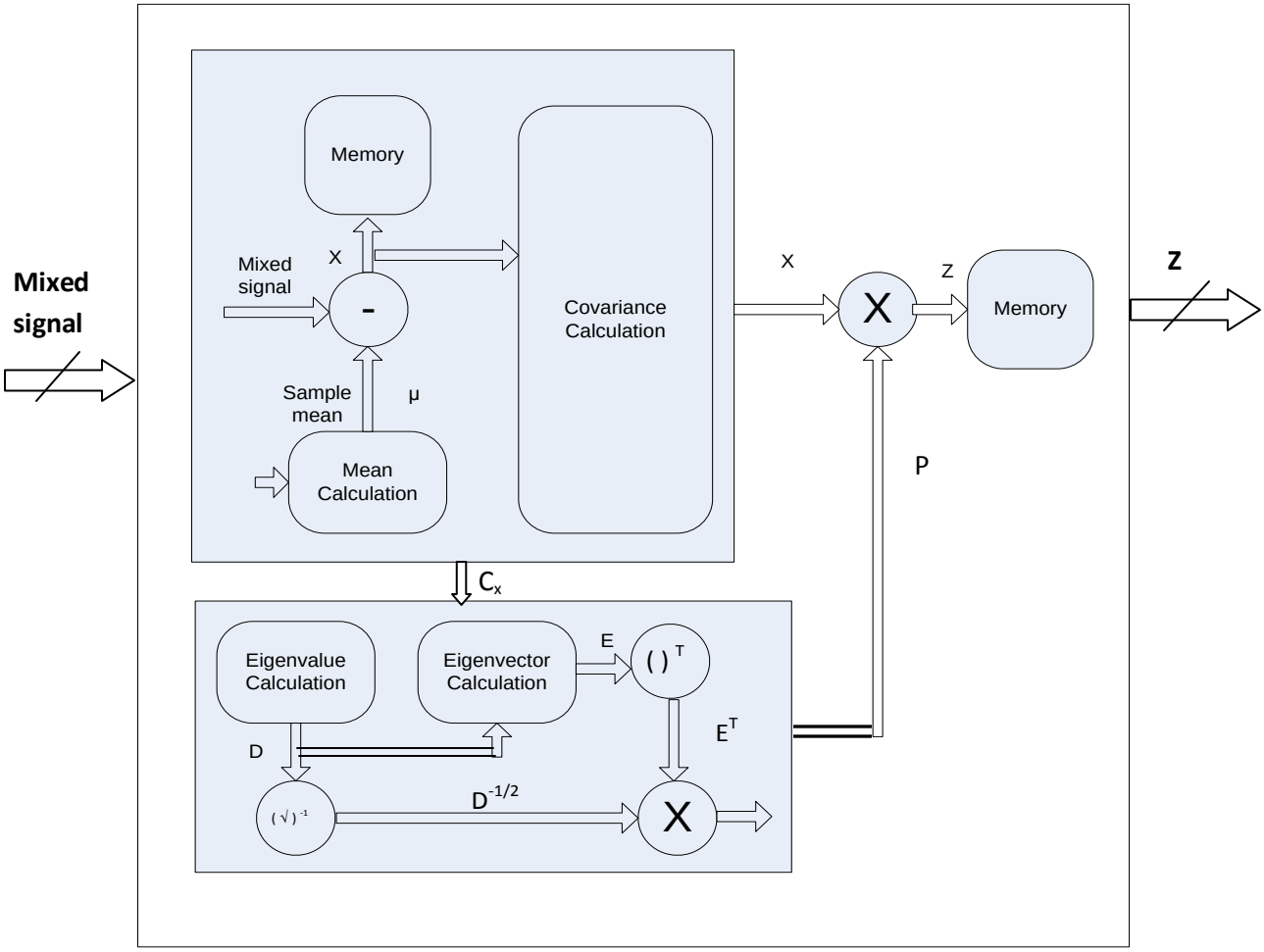

Fig. 5.2 Block diagram of Implementation of Whitening

5.4 Norm Calculation

Initial random vector $w = \begin{bmatrix} W_1 \\ W_2 \end{bmatrix}$ is taken and norm of w is determined.

$$norm\_w = \sqrt{(W_1 \times W_1) + (W_2 \times W_2)} \qquad (5.15)$$

Dividing W_1 and W_2 by its norm

$$w = \begin{bmatrix} w_1 / norm\_w \\ w_2 / norm\_w \end{bmatrix} \quad (5.16)$$

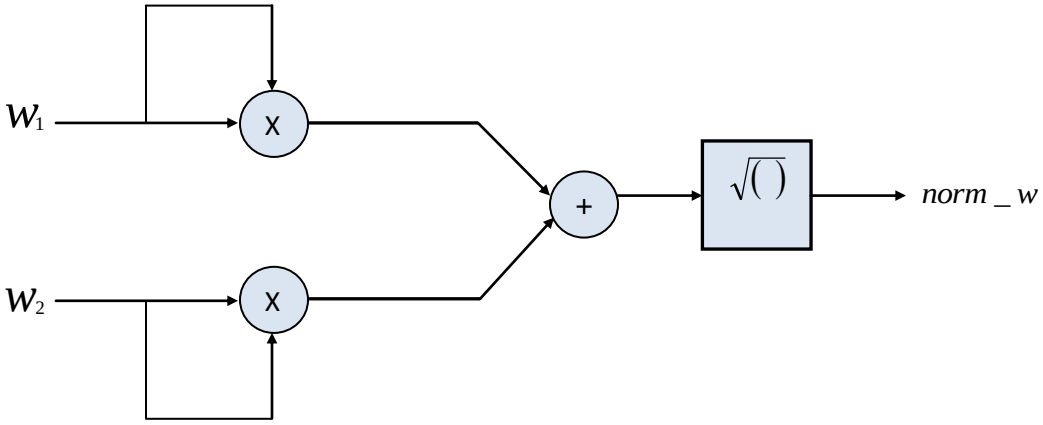

Fig.5.3 Block diagram of norm calculation

5.5 Implementation of Kurtosis

The equation for the calculation of separating vector **w** is expressed as

$$w(k) = E\left\{ Z\left(Z^T \times w(k-1)\right)^3 \right\} - 3 \times w(k-1) \quad (5.17)$$

The calculation of $Z\left(Z^T \times w(k-1)\right)^3$ is first implemented and the concept is presented in Fig 5.4.

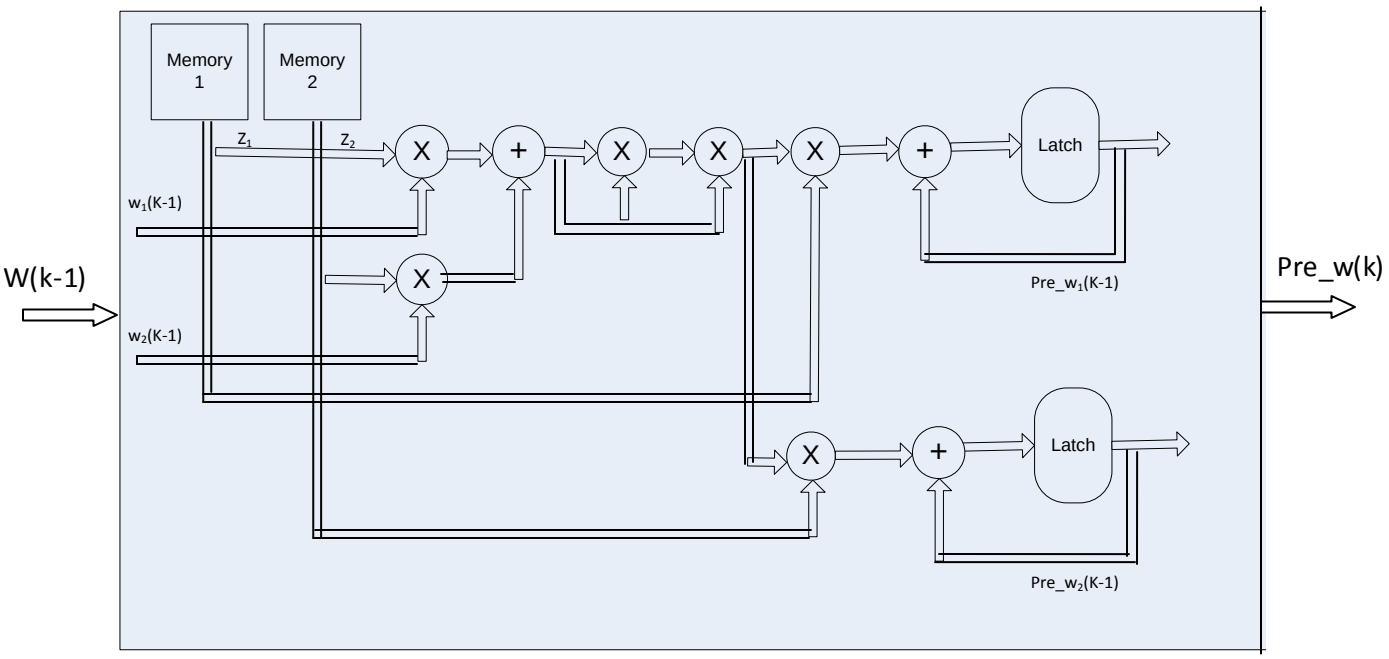

Fig.5.4 Block diagram of Implementation of pre_w(k)

The operation can be formulated as

$$pre\_w(k) = \sum_{j=1}^{samplelength} \left(z(j) \times \left(z(j) \times w(k-1) \right)^3 \right) \tag{5.18}$$

Where pre_w is the calculation result of $Z\left(Z^T \times w(k-1)\right)^3$

The implementation block of $w(k) = E\left\{ Z\left(Z^T \times w(k-1)\right)^3 \right\} - 3 \times w(k-1)$ is presented in Fig.5 .5, where $Z\left(Z^T \times w(k-1)\right)^3 = pre\_w(k) = \left(pre\_w_1(k), pre\_w_2(k)\right)$.

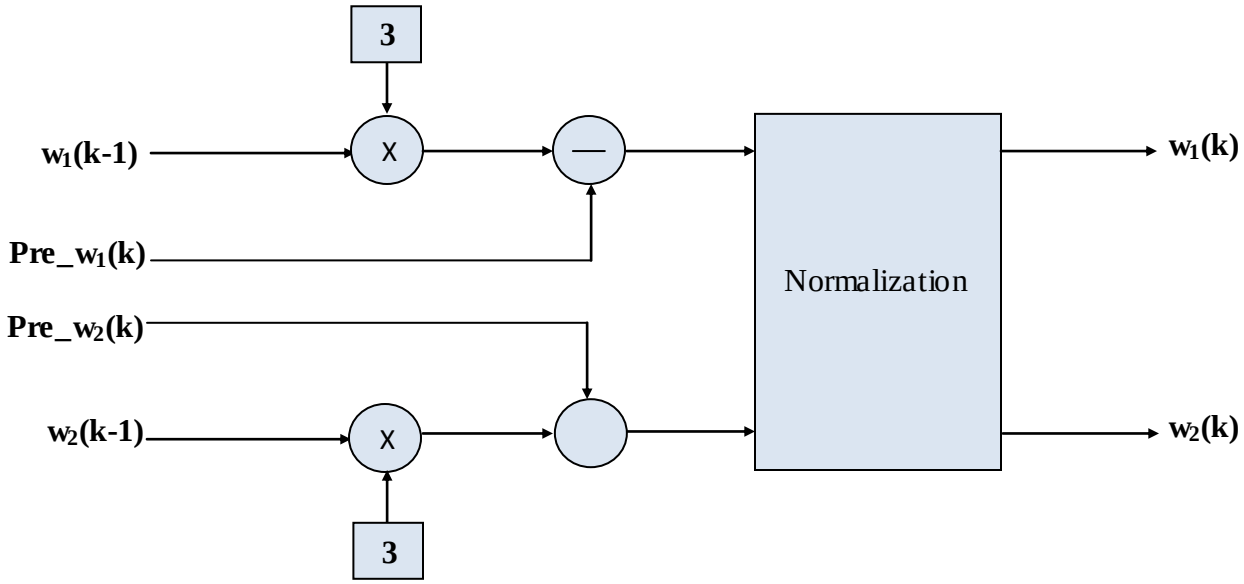

Fig. 5.5 Block diagram of Implementation of w(k)

The normalized value w_new is compared with the old value w_old and if the values do not match then w_new fed back to the input of the block and also stored as w_old in a register for the purpose of comparison. When w_new = w_old then this value is given to the output as the converged vector w which gives one independent component. For finding the other independent component a new random vector w is assumed and it is decorrelated with the earlier w and is again put to the iteration process for getting an optimized converged value

For i=1 and 2 (for two mixed signals)

$$B_1 = \begin{bmatrix} W_1 & 0 \\ W_2 & 0 \end{bmatrix}$$ For the first independent component

$$B_2 = \begin{bmatrix} W_1 & W_{1\_new} \\ W_2 & W_{2\_new} \end{bmatrix}$$ For the second independent component

Where $\begin{bmatrix} W_{1\_new} \\ W_{2\_new} \end{bmatrix}$ are the new random vectors for second independent component

$$B = B_i \times w \qquad (5.19)$$

$$w = w - B \qquad (5.20)$$

Norm of w is again determined.

CHAPTER 6

RESULT AND DISCUSSION

6.1 Simulation of Fast ICA (Using MATLAB)

Simulation 1:

In the simulation, sine and triangular waves generated from MATLAB source are taken as the source signals. Then, the mixed signals are produced by multiplying random mixing matrix and source signals. The sample length of mixed signal X and estimated independent components S_est are both 1000 in the simulation. Fig 6.1(a) and 6.1(b) are the source signals: sine and triangular waves. Fig 6.1(c) and 6.1(d) shows the mixed signals. The demixing matrix B is found by the Fast ICA algorithm and the estimated independent component signals, shown in Fig 6.1(e) and 6.1(f), are derived by multiplying the demixing matrix B and mixed signals.

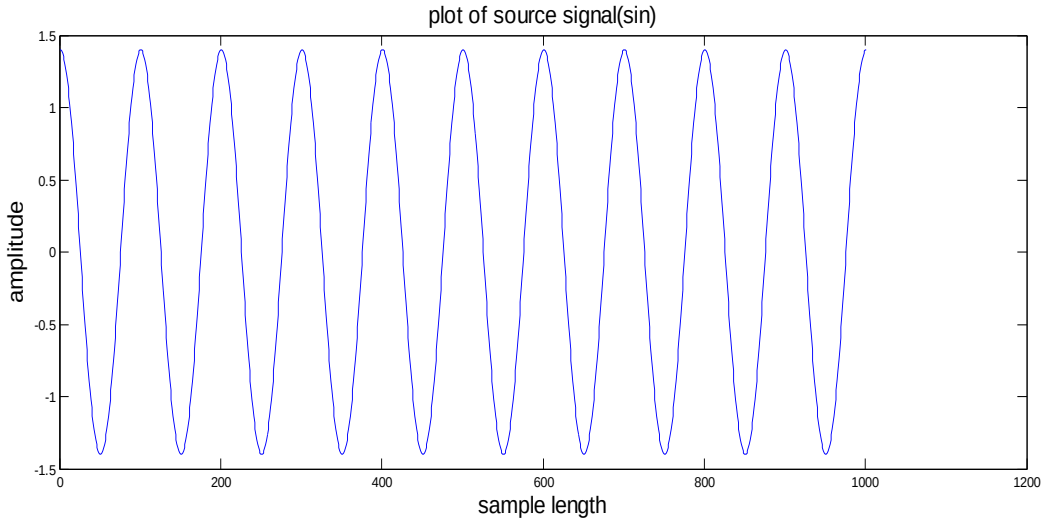

Fig. 6.1(a) Sine Wave (Source signal)

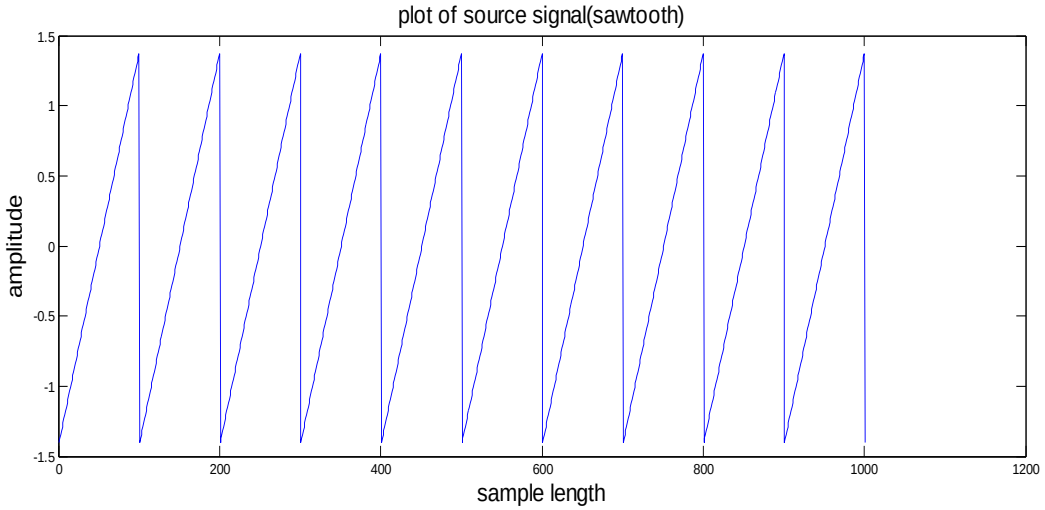

Fig. 6.1(b) Sawtooth Wave (Source signal)

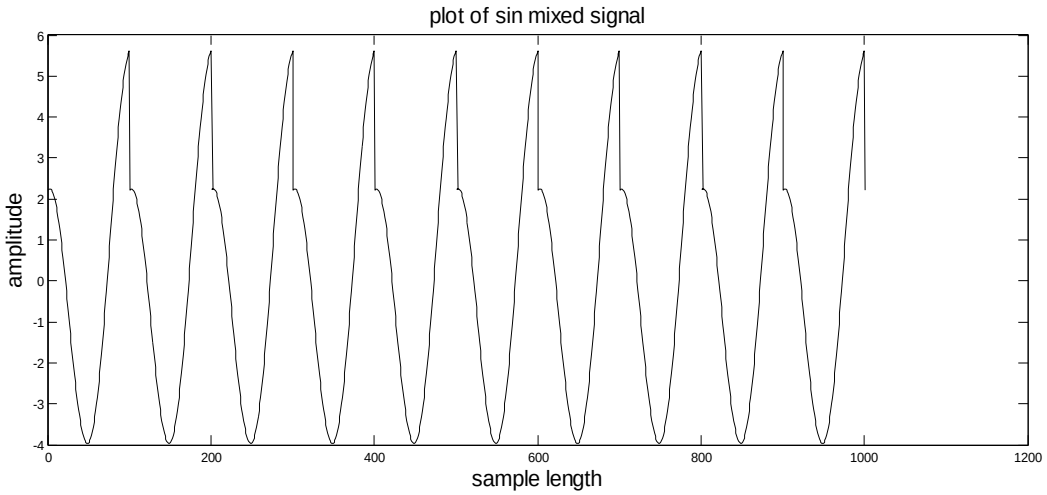

Fig. 6.1(c) Mixed signal 1

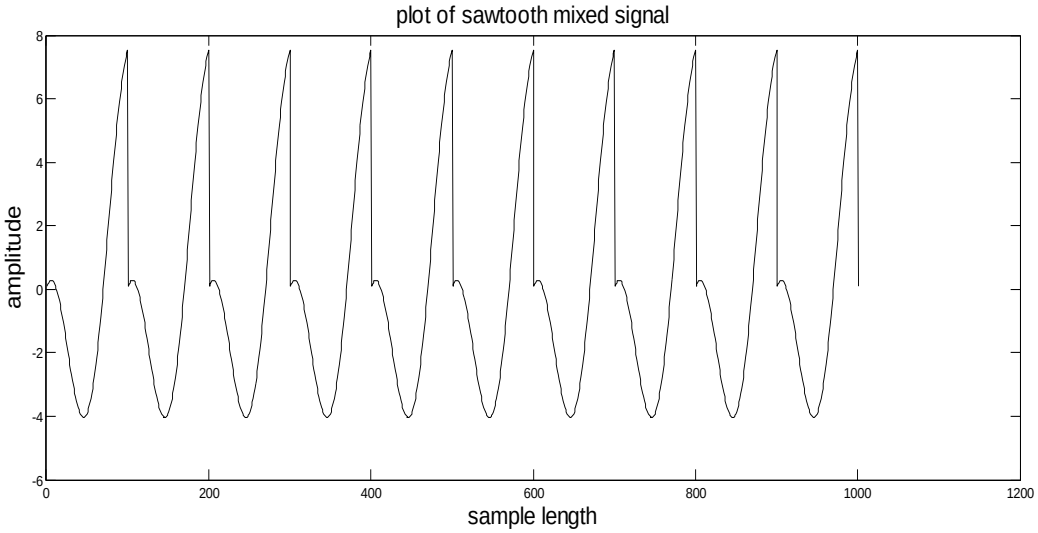

Fig. 6.1(d) Mixed signal 2

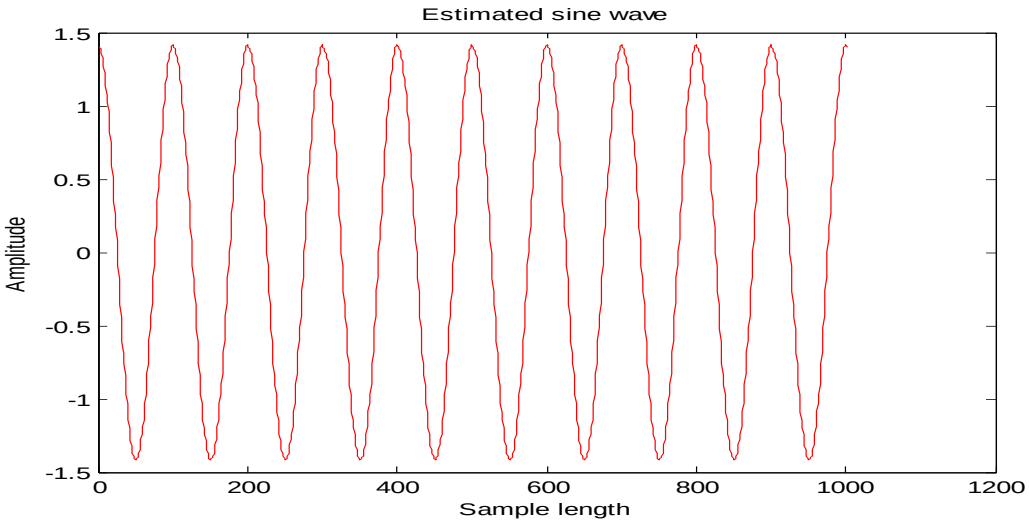

Fig. 6.1(e) Estimated Sine wave

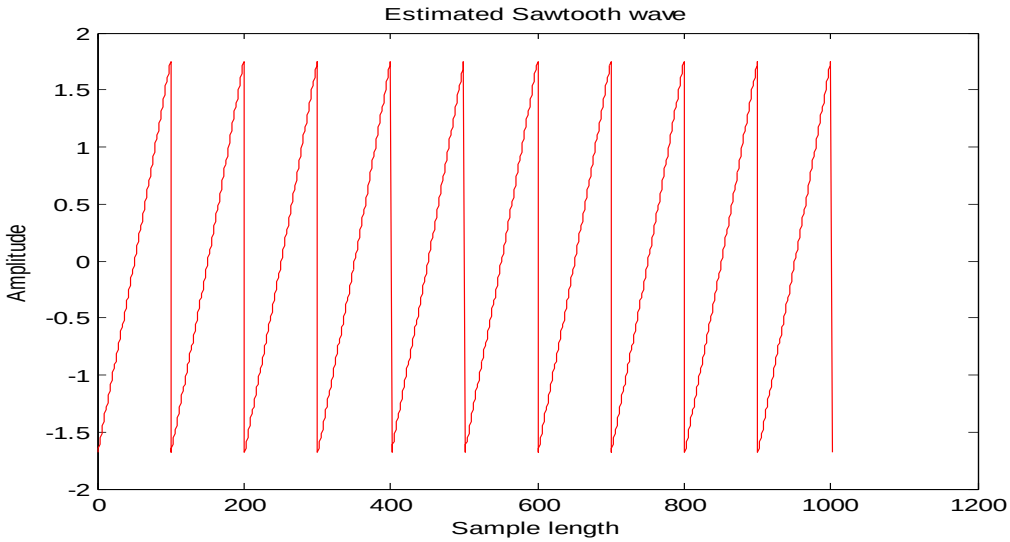

Fig. 6.1(f) Estimated Saw tooth wave

Simulation 2: (Using Real-World Sound)

In this simulation siren signal and speech signal are taken as the source signals. Fig 6.2(a), and 6.2(b) shows the source signals: siren signal and speech signal respectively. The mixed signals shown in Fig 6.2(c) and 6.2(d), are generated by multiplying the randomly generated mixing matrix and the source signals. The estimated independent signals are shown in Fig 6.2(e), and 6.2(f) respectively. Performance comparision, in terms of CPU time , of deflation approach and symmetric approach of the Fast ICA algorithm is carried out taking this example. Table 6.1 shows the comparative result of CPU time for both the approaches.

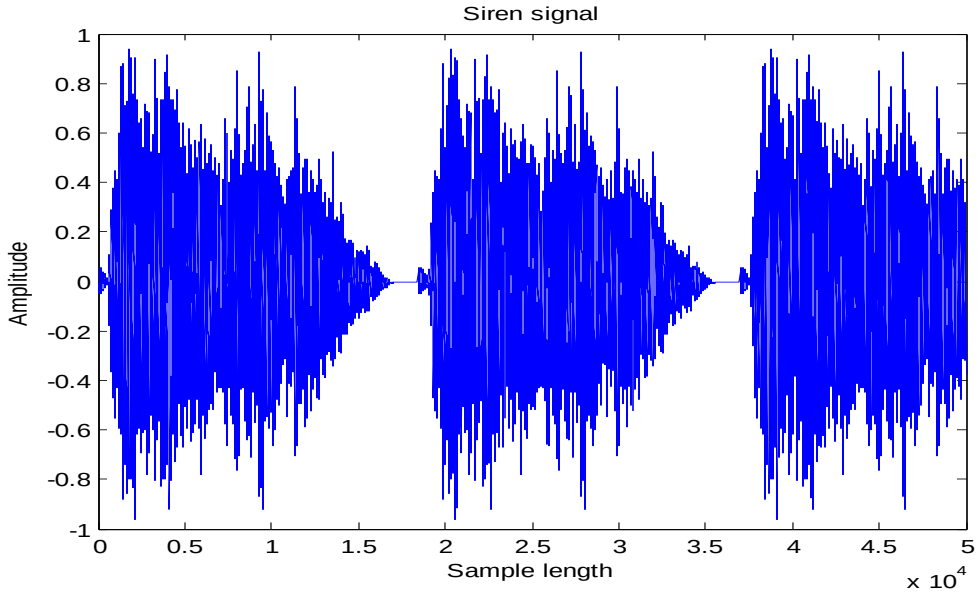

Fig. 6.2(a) Siren signal (source)

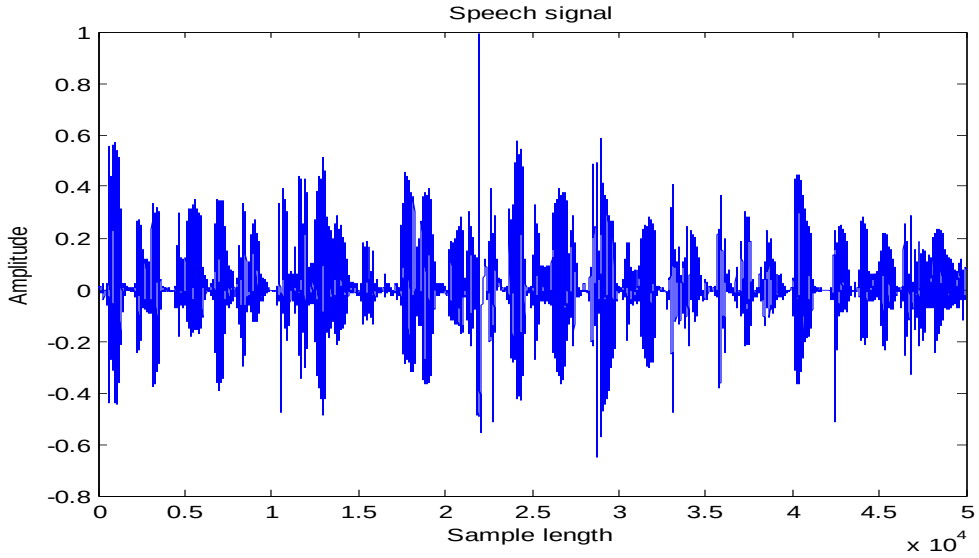

Fig. 6.2(b) Speech signal (source)

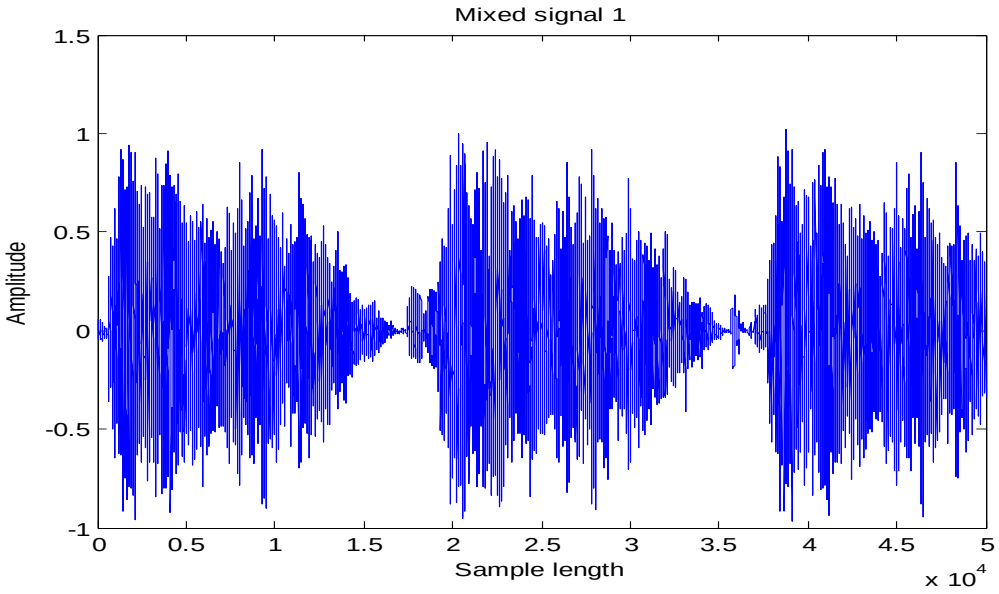

Fig. 6.2(c) Mixed signal 1

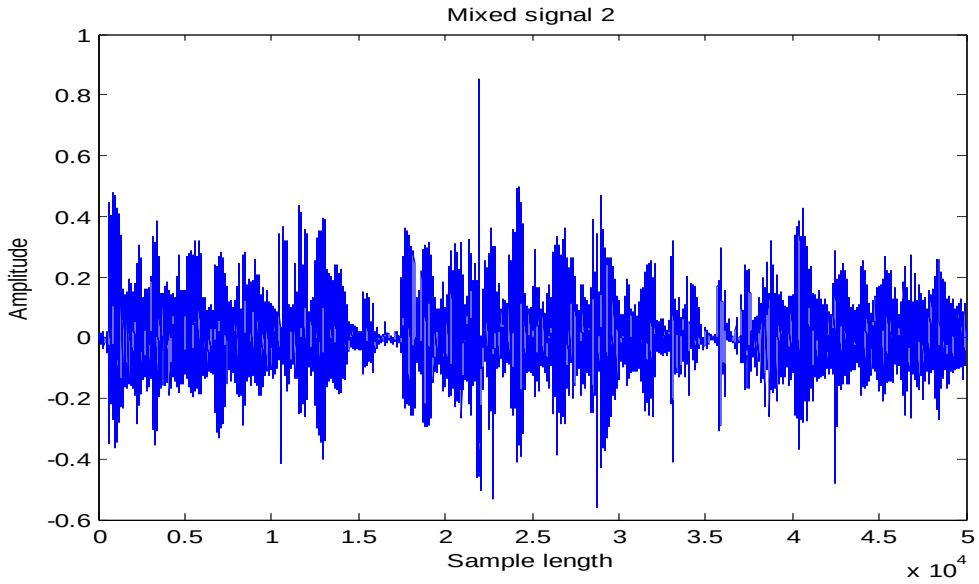

Fig. 6.2(d) Mixed signal 2

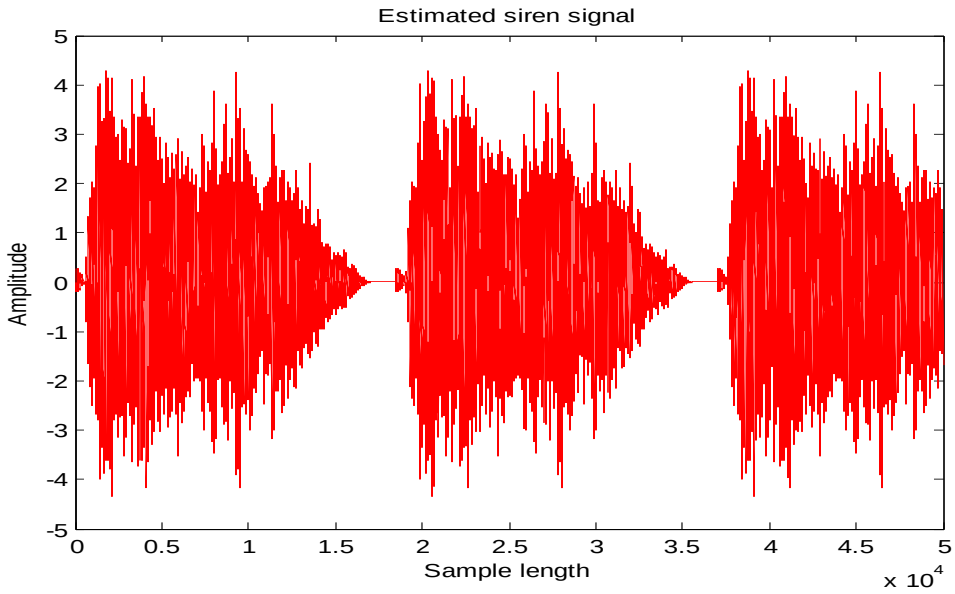

Fig. 6.2(e) Esimated siren signal

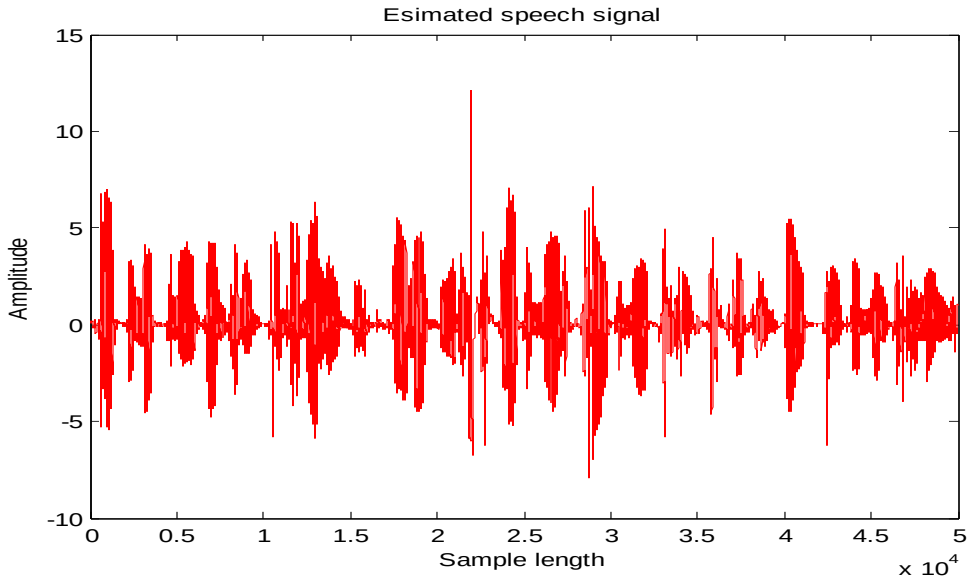

Fig. 6.2(f) Esimated speech signal

FASTICA APPROACH	CPU TIME (in sec)
Deflation approach	146.60
Symmetric approach	129.072

Table 6.1 Comparative result of CPU time by simulating Example 2

6.2 VHDL simulation of Floating point arithmetic

Floating point Adder/Subtractor:

Operation of a floating point adder/subractor block has already been described in previous chapter. Input to this block are two floating point numbers, fp_A and fp_B and the output fp_Y is either the sum or difference of these two numbers depending on the sign bit of the smaller number. Here for addition the two floating point numbers taken are

fp_A = 2 = 0 10000000 00000000000000000000000

fp_B = 5 = 0 10000001 01000000000000000000000

and the output obtained is fp_Y = 7 = 0 10000001 11000000000000000000000

Timing diagram of floating point adder is shown in Fig 6.3

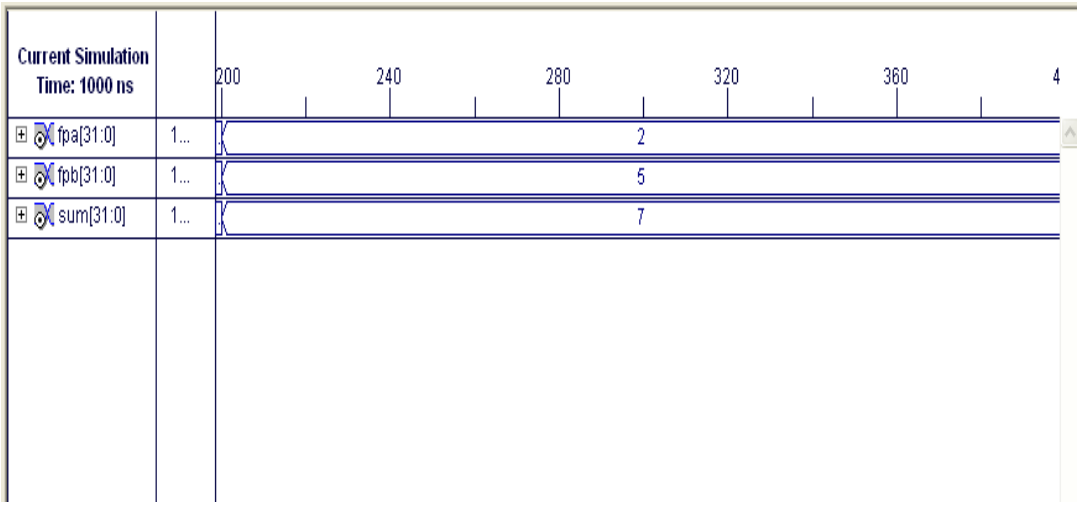

Fig 6.3 Timing diagram of Floating point Adder

For subtraction the two floating point inputs taken are

fp_A = 5 = 0 10000001 01000000000000000000000

fp_B = -2 = 1 10000000 00000000000000000000000

and the output obtained is fp_Y = 3 = 0 10000000 10000000000000000000000

Timing diagram of floating point adder is shown in Fig 6.4

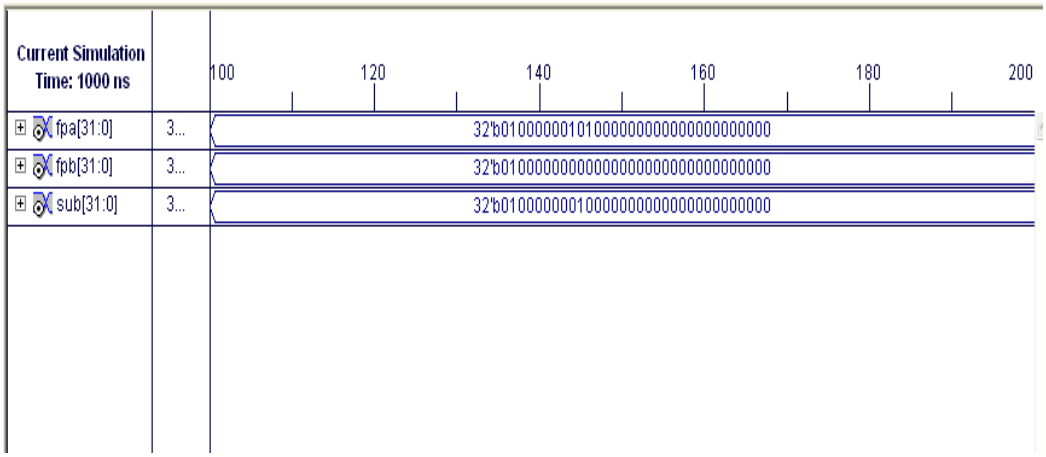

Fig 6.4 Timing diagram of Floating point Subtractor

Floating point Multiplier:

Inputs given to the FP Multiplier block are

fp_A = 2 = 0 10000000 00000000000000000000000

fp_B = 5 = 0 10000001 01000000000000000000000

and the output obtained is prdt_f = 10 = 0 10000010 01000000000000000000000

Fig 6.5 shows the timing diagram of FP multiplication.

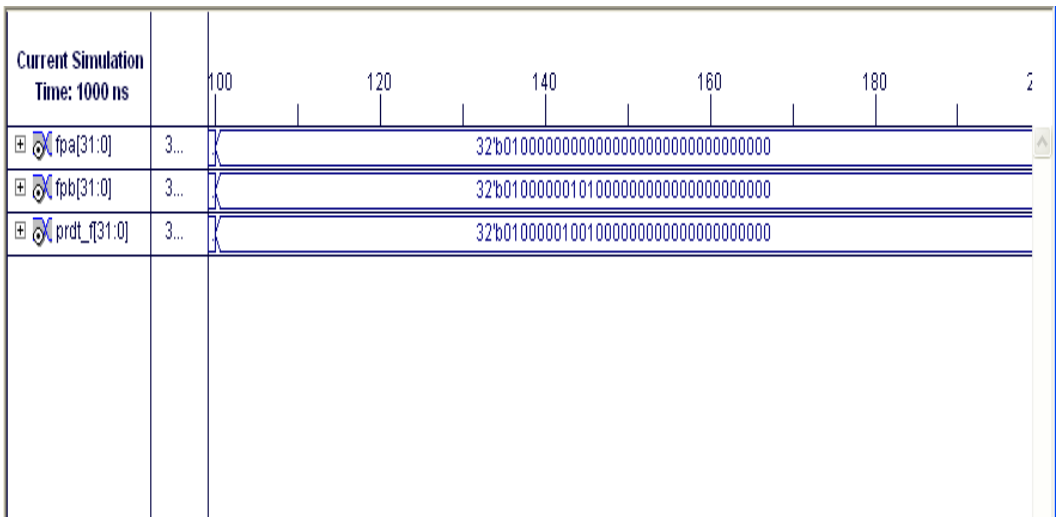

Fig 6.5 Timing diagram of Floating point Multiplication

Floating point Division:

SRT division algorithm is used here to implement the method of 1.fp_A divided by 1.fp_B.

The inputs given to the FP division block are

fp_A = 8 = 0 10000010 00000000000000000000000

fp_B = 2 = 0 10000000 00000000000000000000000

and the output obtained is qout = 4 = 0 10000001 00000000000000000000000

Timing diagram of FP division block is shown in Fig 6.6

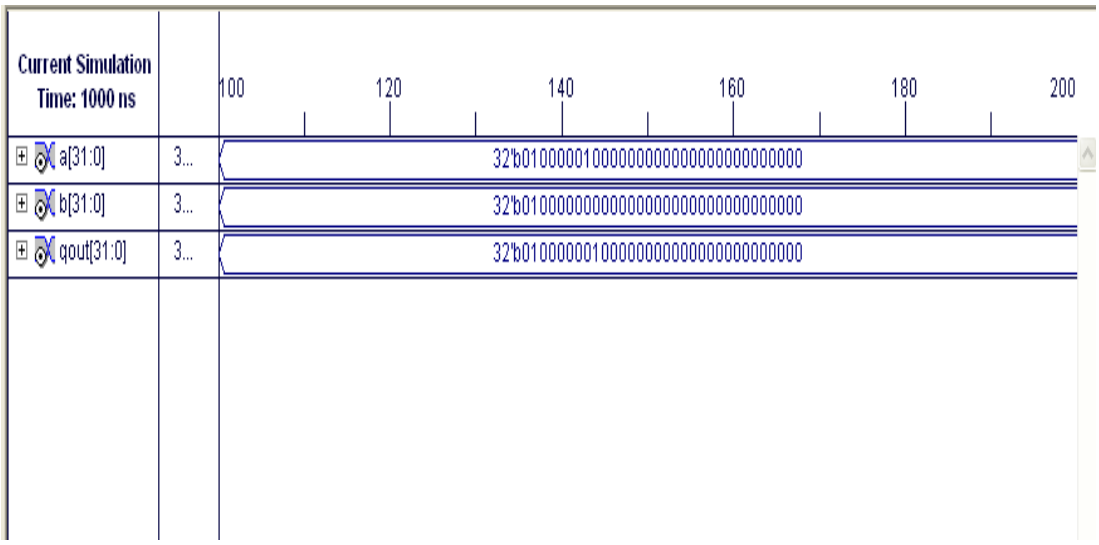

Fig 6.6 Timing diagram of FP division

Floating point Square-Rooter:

Input given to the FP Square-rooter block is *fp* =625 = 0 10001000 00111000100000000000000 and the output of this block i.e. the sruare root of *fp* obtained is

qout = 25 = 0 10000001 10010000000000000000000

Timing diagram of FP square-rooter is shown in Fig 6.7

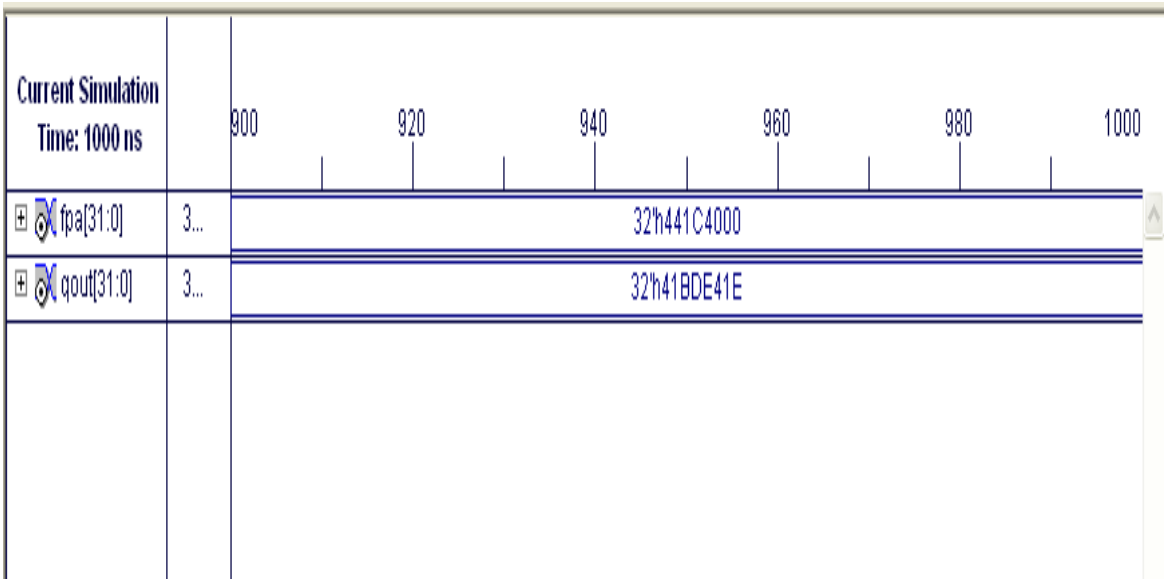

Fig 6.7 Timing diagram of FP Square-rooter

6.3 VHDL Simulation of Different Modules of Fast ICA Algorithm

Centering and Whitening:

Block diagram of implementation of centering and whitening has already been shown in the previous chapter. Input to this block are the mixed signal (X) components and output of this block are the components of the whitened signal (Z),which is produced by multiplying the whitening matrix and the centered mixed signal. Timing diagram of centering and whitening operation is shown in Fig 6.8.

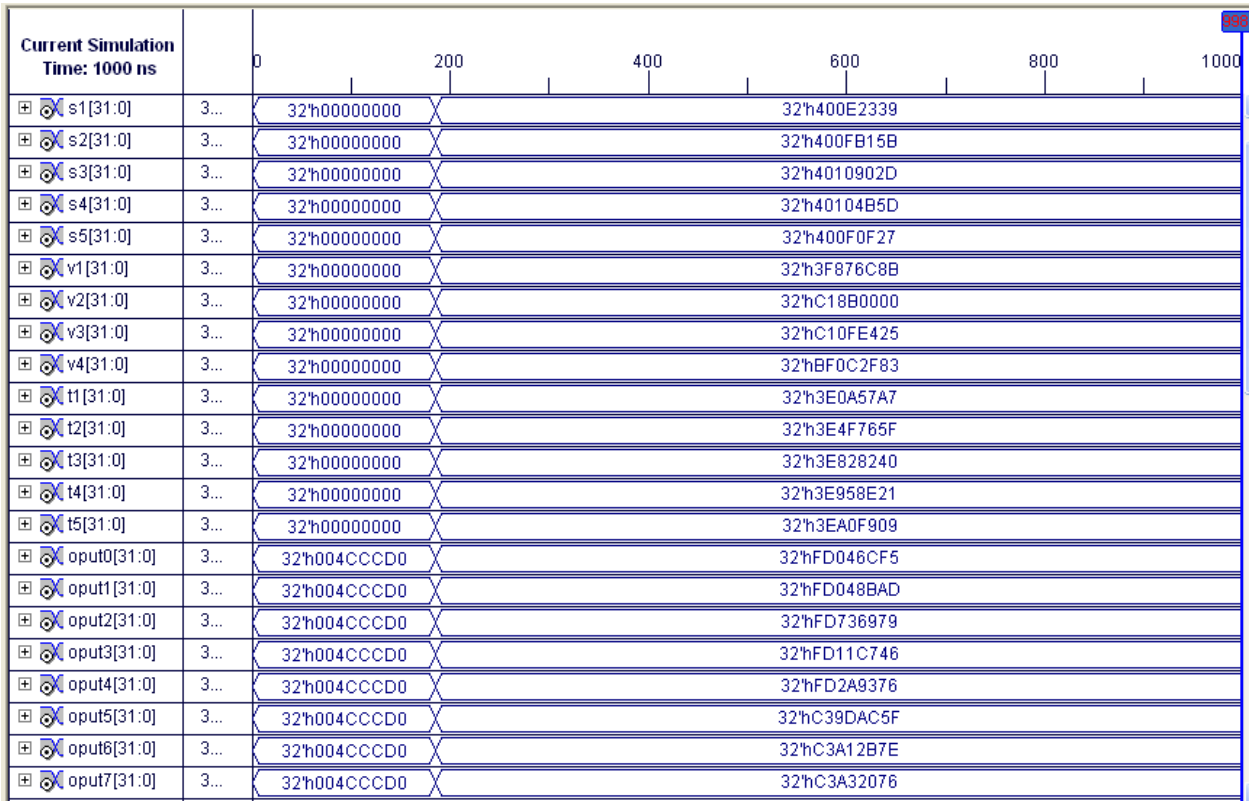

Fig 6.8 Timing diagram of centering and whitening module

Calculation of pre_w (k):

Initially some randomly generated value of $w_1(0)$ and $w_2(0)$ are given as input to this block. These two values operate with the whitened signal components which are given as vector input (z_1 and z_2) to this block and finally gives pre_$w_1(1)$ and pre_$w_2(1)$ as its output. These two value of pre_w are again given as input to another block shown in Fig. 5(e) to produce w(k).Then the corresponding values of w(k) are given as input to this block to get the corresponding pre_w_1(k) and pre_w_2(k).Timing diagram of this module is shown in Fig 6.9.

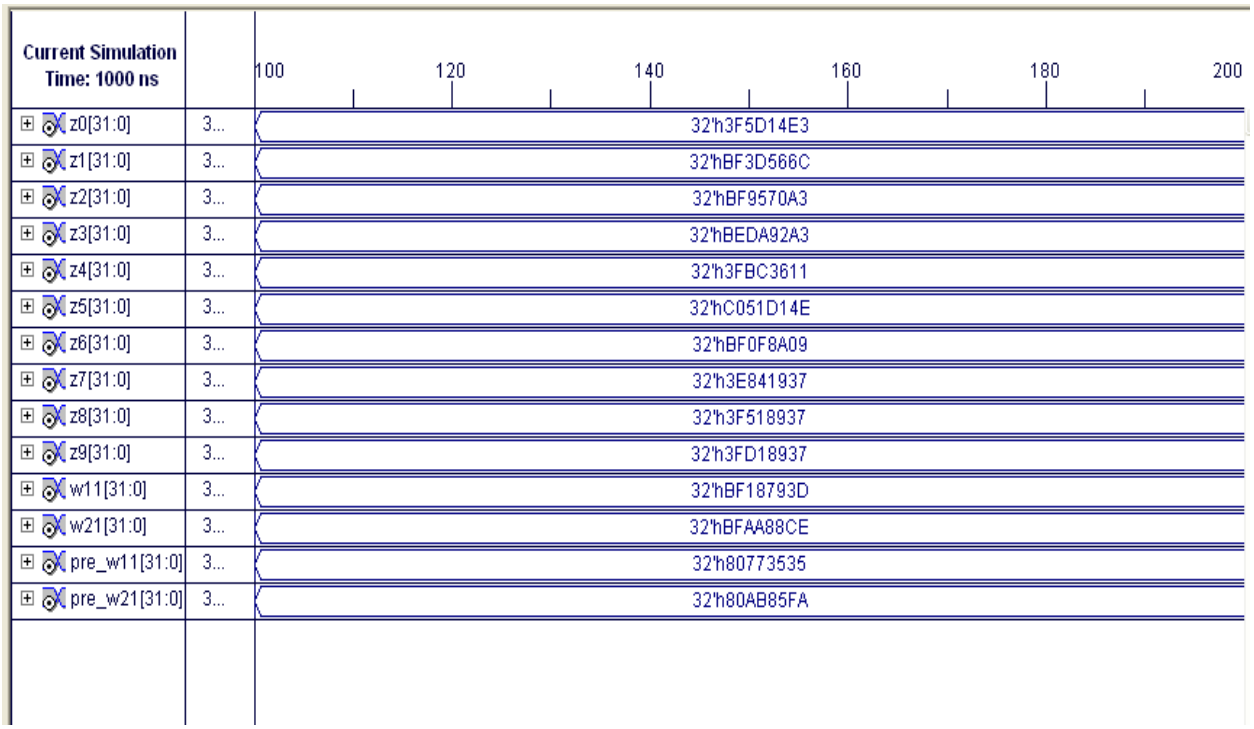

Fig 6.9 Timing diagram of Pre_w calculation Module

Calculation of w (k):

Once the calculation of pre_w(k) is over, the next step is to determine the new value of w(k). Block diagram of implementation of this operation has already been shown in the previous chapter. Fig 6.10 shows the timing diagram of this block. Input to this block are pre_w_1(k) and pre_w_2(k) which have already been determined in the previous module and the output of this block are the new values of w_1(k) and w_2(k) which are then normalized.

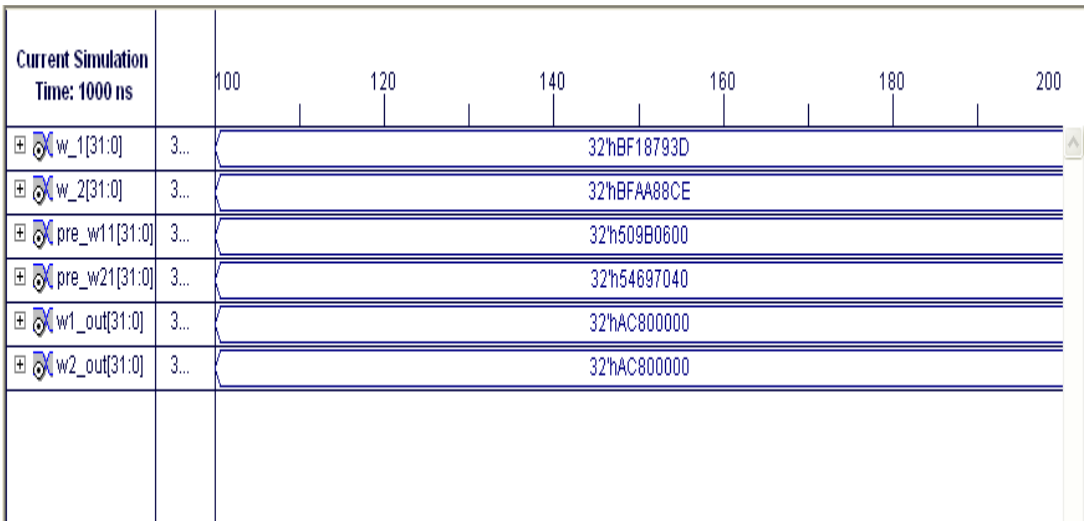

Fig 6.10 Timing diagram of w(k) calculation Module

CHAPTER

7

CONCLUSION

7.1 CONCLUSION

Ways needed in fast ICA algorithm for decorrelation of the separating matrix can be deflationary or symmetric orthogonalization. In some applications, it may be preferable to use the fast ICA algorithm with symmetric orthonormalization, in which every vector is impartially treated and the parallel computation of independent components is enabled. Extensive simulation studies reveal that symmetric approach has a better performance as compared to deflation approach, in terms of CPU time. The 32 bit floating-point (FP) arithmetic is implemented by hand coding HDL code to provide better precision and higher dynamic performance. VLSI implementation of Fast ICA algorithm offers many features such as high processing speed, which is extremely desired in many applications. In order to reduce the complexity, the Fast ICA block is divided into several sub modules and each of the sub modules are developed by HDL coding.

7.2 SCOPE FOR FUTURE WORK

The proposed research can be extended in following dimensions.

A pipeline architecture of the Fast ICA algorithm can be developed for real-time sequential mixed signal processing. An extended implementation of Fast ICA algorithm based on the proposed modules can be done for higher dimension (more than two souces and mixtures).VLSI implementation of different ICA technique can be carried out.

REFERENCES

[1]. Kuo-Kai Shyu,Ming-Huan Lee, Yu-Te Wu, Po-Lei Lee " Implementation of pipelined FastICA on FPGA for Real-time Blind source separation". IEEE Trans. Neural Netw.,vol.19,no.6,June 2008.

[2]. J.T. Chien and B.C. Chen, " A new independent component analysis for speech recognition and separation," IEEE Trans. Speech Audio Process.,vol.14,no.4,pp.1245-1254,July 2006.

[3]. C. M. Kim, H. M. Park, Y. K. Choi, and S. Y. Lee, "FPGA implementation of ICA algorithm for blind source separation and adaptive noise cancelling," IEEE, Trans, Neural Netw.,vol.14,no.5,pp.1038-1046,Sep.2003.

[4]. A. Hyvarinen, J. Karhunen, and E. Oja, " Independent component analysis: Algorithms and applications," Neural Netw.,vol.13.,pp.411-430,May 2000.

[5]. Changyuan Fan, Baoqiang Wang,Hui Ju,"A new FastICA algorithm with symmetric orthogonalization" Communications, circuits and system proceedings,vol.3,pp.2058-2061, June 2006.

[6]. P. Comon, "Independent Component Analysis-A new concept?" Signal Processing, vol-36, pp.287-314, 1994.

[7]. J.F.Cardoso, "Blind Signal Separation: Statistical Principles", Proc. of IEEE, vol-9, No.10, pp. 2009-2025, 1998.

[8]. Scott C. Douglas"A statistical convergence analysis of the fastica algorithm for two-source mixtures", IEEE Trans. Neural Netw., vol. 14, pp. 943-949,July 2003.

[9]. Abdullah Celik, Milutin Stanacevic and Gert Cauwenberghs"Mixed-signal real-time adaptive blind source separation",Circuits and systems,2004.ISCAS '04.Proceedings of the 2004 Inteernational symposium on Volume 5, 23-26 May 2004 Page(s):V-760 - V-763 Vol.5

[10]. HongLi, Yunlian Sun, "The study and test of ICA algorithms "Wireless Communications,Networking and mobile computing,2005. Volume 1, 23-26 Sept. 2005 Page(s):602 – 605.

[11]. E.Oja, "Nonlinear PCA criterion and maximum likelihood in independent component analysis", Proc. Int. Workshop on Independent Component Analysis and Signal separation (ICA'99), pp.143-148, Aussois, France, 1999.

[12]. A. Hyvarinen, J. Karhunen and E. Oja, " Independent Component Analysis". John Wiley & Sons, Inc., New York, 2001.

[13]. A. K. Nandi and F. Herrmann, "Fourth-order cumulant based estimator for independent component analysis", Electronic Letters, 37(7), pp.469–470, 2001.

[14]. A.Hyvarinen, " Fast and robust fixed-point algorithms for independent component analysis". IEEE Trans. Neural Netw.,vol.10,no.3,pp.624-634,May 1999.

[15]. A. Hyvarinen and E. Oja, "A Fast Fixed-Point Algorithm for Independent Component Analysis," *Neural Computation, vol. 9, pp.*1483-1492, 1997.

[16]. H.Du,H.Qi and X.Wang, "Comparative Study of VLSI Solutions to Independent Component Analysis", IEEE Trans. Industrial Electronics, vol.54, No.1, 2007.

[17]. Patel, J.N.,Abid Z., Wang W,"VLSI implementation of a floating-point divider",Microelectronics,2004.ICM.2004 proceedings.

[18]. R.P.Brent and F.T.Luk, "The solution of singular-value and symmetric eigen-value problems on multiprocessor arrays", SIAM J. Scientific and Statistical Computing, 6, pp.69-84, 1985.

[19]. H.Du,H.Qi and X.Wang, "Comparative Study of VLSI Solutions to Independent Component Analysis", IEEE Trans. Industrial Electronics, vol.54, No.1, 2007.

[20]. N. Shirazi, A. Walters and P. Athanas, "Quantitative analysis of floating point arithmetic on FPGA based custom computing machines," in Proc. IEEE Symposium on FPGAs for Custom Computing Machines, 1995, pp. 155-162.

[21]. Yamin Li and Wanming Chu "A New Non-Restoring Square Root Algorithm and Its VLSI Implementations"